UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA DE MONITORIZACIÓN DE REDES SCADA PARA LA DETECCIÓN DE TRÁFICO ANÓMALO

Álvaro Culebras Sánchez
Tutor: Jorge E. López de Vergara Méndez

ENERO 2016

Desarrollo de un sistema de monitorización de redes SCADA para la detección de tráfico anómalo

AUTOR: Álvaro Culebras Sánchez
TUTOR: Jorge E. López de Vergara Méndez

High Performance Computing and Networking Research Group

Dpto. Tecnología Electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Enero de 2016

Resumen

Las redes SCADA (Supervisory Control and Data Acquisition) constituyen uno de los sistemas de control de datos en entorno industrial más extendidos hasta el momento. Dichas redes se encargan de la supervisión, control y adquisición de datos de relevancia en procesos industriales controlados, como aquellos que forman parte de la producción energética o tratamiento de recursos para el beneficio de los humanos. Durante el último lustro el número de amenazas a la seguridad en la red ha aumentado exponencialmente hasta convertirse en uno de los principales problemas para empresas u organizaciones, tanto públicas como privadas. Las redes SCADA no son una excepción, siendo de especial importancia el despliegue de medidas de seguridad que permitan el correcto funcionamiento de los procesos que controlan. La tendencia actual a la interconexión de la mayoría de los sistemas desplegados o la gestión de incidencias de manera remota provoca la aparición de múltiples vulnerabilidades, fallos de seguridad y filtraciones en los datos o recursos de cualquier ente. En concreto, es de estudio de este documento la seguridad en el entorno de un sistema industrial de supervisión, control y adquisición de datos (SCADA) sobre el protocolo ModbusTCP, así como el rendimiento logrado a través de las soluciones propuestas para la detección de tráfico anómalo. Estas soluciones constituyen en su conjunto un sistema de detección de intrusiones (IDS), cuya misión es la de alertar sobre las posibles amenazas que afecten al entorno SCADA. Este tipo de redes se caracterizan por tener una baja tasa de intercambio de paquetes junto a una muy alta regularidad en el intervalo de las peticiones al servidor, lo cual permite diseñar soluciones de detección de amenazas en tiempo real una vez lograda la reproducción de la caracterización de la red. El sistema IDS se encarga de la captura de los paquetes sin necesidad de redirigir el tráfico, del análisis de estos paquetes mediante el estudio de las características habituales en los protocolos presentes habitualmente y de la posterior generación de estadísticas y ratios necesarios para el lanzamiento de avisos por tráfico anómalo.

Palabras clave

Ciberseguridad, SCADA, IDS, ModbusTCP, TCP, detección por anomalía, seguridad industrial

Abstract

Nowadays, industrial supervisory, control and data acquisition (SCADA) networks constitutes one of the most extended industrial data control systems. These networks acquire and supervise relevant data from industrial processes such as some related with energy production or treatment of human resources. During these last five years the number of cybersecurity attacks has increased exponentially until becoming one of the biggest problems for both, enterprise and public organisms. It is therefore included as one of the most important issues on the agenda for many of the organisms mentioned before. The trend to interconnect deployed systems or to manage any trouble that appears on the system remotely provokes the growth of multiple vulnerabilities, security breaches and information leaks from any organization. Specifically, this document studies the security of an SCADA network over ModbusTCP protocol as well as the performance achieved by the developed solutions for the anomalous traffic detection. These solutions conform what is usually named as an Intrusion Detection System (IDS), whose mission is to alert the system manager about the possible threats that can affect SCADA. This kind of network is characterized by having low packet transmission rates and high deterministic petition intervals that allows the design of real-time detection solutions after identifying how the protocol works. This IDS is responsible of capturing packets without the need to redirect the packet flow, analyzing these packets by means of the usual ModbusTCP protocol characteristics and generating statistics and ratios to correctly show warnings and alerts.

Keywords

Cybersecurity, SCADA, IDS, ModbusTCP, industrial security, TCP

Agradecimientos

Siempre es difícil reunir en pocas líneas los agradecimientos a todas las personas que han sido partícipes en mayor o menor medida de mi paso por la Universidad, con todo lo que ello conlleva.

En primer lugar, me gustaría agradecer a toda mi familia el apoyo que me han brindado a lo largo de estos años, en los buenos y malos momentos, aguantando las largas jornadas de estudio y progreso de este Trabajo de Fin de Grado.

En segundo lugar, agradecer a mi tutor Jorge E. López de Vergara la ayuda, conocimientos y ánimos ofrecidos durante los meses de realización del TFG, sin los cuales ninguna de las páginas de este documento sería posible. Gracias por tantas preguntas planteadas y tantas respuestas ofrecidas.

También, en la medida en la que les corresponde, debo agradecer a todos aquellos profesores que en algún momento de mi paso por esta Escuela prestaron un instante de su tiempo para ofrecer nuevos conocimientos y puntos de vista. De algunos de ellos guardo un grato recuerdo por el apoyo que brindaron en los peores momentos.

En el plano personal, quiero aprovechar para acordarme de gente que, siendo consciente más o menos de su apoyo, han hecho más fácil cada instante de la vida universitaria: Pali, Manuel, Alejandro, Paula, Ángel, Tomé, Erik, Cobos, Edu. De cada uno de vosotros he aprendido algo que me gustaría guardar para siempre. Más allá del aspecto universitario, no puedo dejar de apreciar el entendimiento y apoyo de amigos como Antonio, Pablo, Toni, Edu, Ernesto, Maya, Susana y María, que perdonaron mis ausencias y celebraron los éxitos como propios.

Finalmente, el mayor de mis agradecimientos va dedicado a la mejor compañera de viaje, confidente, apoyo en los malos momentos e impulso extra en cada instante especial de mi vida. Bea, sin ti no sería lo mismo.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

INDICE DE TABLAS

GLOSARIO DE TÉRMINOS

SCADA: Supervisory Control and Data Acquisition
TCP: Transmission Control Protocol
IP: Internet Protocol
IDS: Intrusion Detection System
IPS: Intrusion Prevention System
PLC: Programmable Logic Controller
DDOS: Distributed Denial of Service
SPA*:* Stateful Protocol Analysis
MODBUSTCP: Modbus Over TCP
MBAP: Modbus Application Protocol Header
ACK: Acknowledgment
PSH: Push
FIN: Finalization
SYN: Synchronization

1 Introducción

1.1 Motivación

El aumento de las infraestructuras conectadas a Internet y la gestión de éstas en modo remoto han producido un gran avance en la eficacia a la hora de controlar procesos industriales. Los avances tecnológicos permiten a las empresas e infraestructuras del estado mejorar la calidad de los productos ofrecidos a la vez que se optimizan los recursos tanto materiales como humanos.

Éste mismo punto fuerte de eficiencia y uso contenido de recursos permite la aparición de vulnerabilidades en el proceso de gestión de cualquier red con acceso a internet siendo igualmente sencillo que un usuario de la red con hábiles conocimientos informáticos y herramientas adecuadas ponga en jaque la seguridad de los datos alojados o transferidos por esta vía.

Si se analizan los datos referentes a ciberataques a infraestructuras críticas en España [1], se obtiene un número aproximado 172 episodios de cibercrimen relacionados con infraestructuras industriales o intelectuales.

Al encontrarse en una situación en la que un ciberataque a una red SCADA (*Supervisory Control and Data Acquisition*) supone en cuanto a costes un esfuerzo mínimo frente a las pérdidas ocasionadas por un funcionamiento errático o la parada de los procesos habituales de un entorno industrial, se percibe necesaria la toma de medidas de seguridad que permitan la prevención de dichas amenazas.

Es por ello que la detección de tráfico anómalo en una red a la que pertenece una red SCADA, la cual controla y comunica a los clientes datos de índole industrial (control de procesos, valores térmicos, químicos, etc.), se vuelve primordial para evitar fugas de información o lo que sería aún más grave: alteración no autorizada de datos u órdenes dadas al sistema ilegítimamente mediante un acceso remoto [2] no autorizado.

En este contexto se desarrolla la idea de la que parte la realización de este Trabajo de Fin de Grado: el diseño y prueba de una herramienta de detección de tráfico anómalo en redes SCADA con el fin de alertar de la presencia de posibles amenazas para la seguridad de datos de procesos industriales.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado es el diseño e implementación de un sistema de detección de intrusiones, denominado IDS o Sistema de Detección de Intrusiones.

Un sistema de estas características permite sondear la red en la que se sitúe, ya sea embebido en un *appliance* de seguridad o mediante su despliegue como software adjunto a un servidor. Para ello se realiza la captura de paquetes en el lado de conexión del servidor SCADA. Se disecciona y estudia cada uno de los paquetes recibidos por el puerto habitual de conexión de ModbusTCP (502), prestando atención especial a los códigos de función, *flags* TCP e IP, número de paquetes por segundo y tamaño medio de paquete por segundo.

Este desarrollo, que en publicaciones previas a este Trabajo de Fin de Grado denominan "detección por anomalía" [3] [4] permite el establecimiento de unos valores máximos y mínimos esperados ante los cuales el IDS reacciona en función del valor actual calculado. De la misma manera, y con el fin de ampliar las posibilidades y generar otros métodos de detección que permitan en futuros trabajos el diseño de estrategias de detección denominadas "detección por firma" e incluso facilitar el desarrollo de técnicas de *clustering* aplicadas a este entorno [5], se generan una serie de estadísticas suplementarias que pueden ser mejoradas y exportadas a las estrategias anteriormente definidas.

Las directrices principales para este IDS son las siguientes:

- Creación de una aplicación con capacidades de análisis en tiempo real que permita detectar las posibles amenazas en la red.
- Generación de estadísticas que permitan lanzar avisos o alertas.
- Monitorización y caracterización de un sistema SCADA, dado que no se tienen referencias previas en la Escuela.
- Medición del rendimiento de la solución en las situaciones a las que se someta a la aplicación.

1.3 Fases de realización del Trabajo de Fin de Grado

El desarrollo de un IDS para un sistema SCADA se considera pionero en nuestra Escuela, por lo que fue necesaria la búsqueda previa de sistemas que permitiesen recrear las condiciones dadas en una red de este tipo. Las fases que se han seguido en la realización del trabajo son las siguientes:

- Documentación: Durante los meses finales del curso 2013/2014, los meses de verano y primeros meses del curso 2014/2015 fue necesaria la lectura de publicaciones que relacionaban los tipos de detección habituales, el funcionamiento habitual de un sistema SCADA y la estructura fundamental del protocolo ModbusTCP, así como una actualización de conocimientos relacionados con las redes basadas en protocolos Ethernet, IP y TCP.

- Diseño de entorno para pruebas: Dada la particularidad del sistema estudiado, la realización desde cero de un entorno funcional y suficientemente ajustado a la realidad de estos sistemas supuso un reto y un valor añadido gracias a la experiencia adquirida en

pruebas con diferentes distribuciones Linux, sistemas de instalación de librerías y paquetes, sistemas de virtualización e interconexión de máquinas virtuales, etc.

• Análisis: Con el entorno de pruebas ajustado y los sistemas esclavo y maestro configurados correctamente en una red que más tarde se procede a caracterizar el flujo de tráfico habitual entre un servidor o esclavo SCADA y un peticionario llamado maestro o cliente. Este paso permite atisbar y acotar en un primer acercamiento algunos posibles umbrales de detección para nuestro IDS.

• Desarrollo de la solución: Esta etapa constituye el grueso de la realización del Trabajo de Fin de Grado. Se pueden diferenciar diferentes fases dentro del IDS, siendo desarrollada en primer lugar la fase de captura y disección de paquetes y posteriormente la fase de generación de estadísticas y umbrales de detección.

• Validación: Durante las fases finales del desarrollo de la aplicación y a continuación del fin del desarrollo se establecen una serie de métricas con el fin de conformar un banco de pruebas múltiples, capaz de analizar por separado el rendimiento de la fase de captura y disección y el rendimiento de la fase de generación de estadísticas y umbrales.

• Redacción de la memoria: Como uno de los pasos finales, se redacta el documento que se muestra con la finalidad de plasmar y demostrar el conocimiento adquirido durante este año en la realización del Trabajo de Fin de Grado.

• Presentación: Finalmente, se realiza una presentación que aúna y resume los principales objetivos y características para la exposición final del trabajo.

Con el fin de resumir gráficamente el progreso realizado durante la realización de este Trabajo de Fin de Grado se incluye el siguiente Diagrama de Gant.

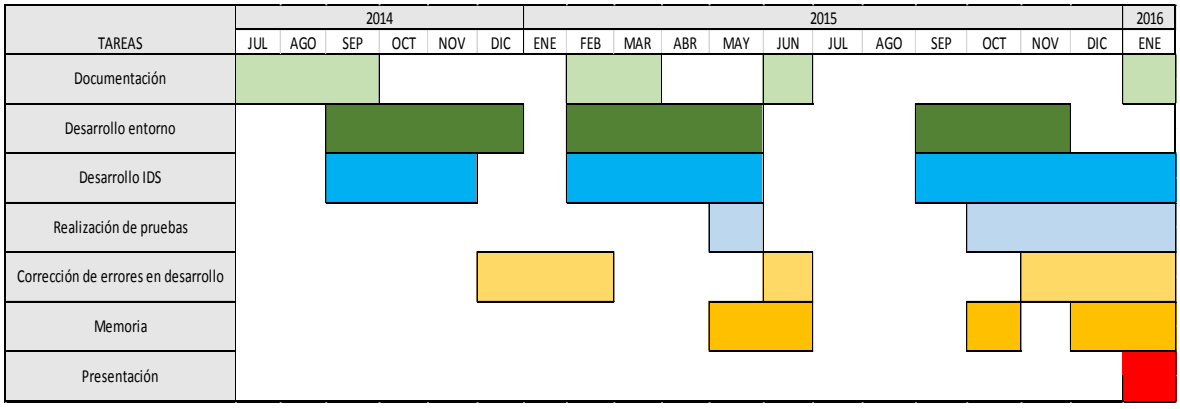


Figura 1-1 : Cronograma de realización del Trabajo de Fin de Grado

1.4 Estructura del documento

En este apartado se detalla la estructura del Trabajo de Fin de Grado:

1. En el segundo capítulo se realiza una exposición del Estado del Arte en cuanto a ciberseguridad y más específicamente sobre IDS dedicados a SCADA. Así mismo se ofrece un pequeño resumen de los protocolos más habituales en la transmisión de información de estos sistemas SCADA. Además se detallan y explican las condiciones que nos llevan a escoger el protocolo ModbusTCP y se ofrece además un extenso análisis de las características de este protocolo, así como su funcionamiento habitual.

2. En el tercer capítulo se explican el funcionamiento y requisitos necesarios y/o deseados que debe cumplir la solución de ciberseguridad desarrollada. Se detallan de manera técnica las funcionalidades deseadas y se relacionan con las técnicas usadas en otras publicaciones. Además se profundiza en el uso eficiente de la información que nos proporciona ModbusTCP y las peculiares características de una red de tasa baja de transmisión como la que nos concierne.

3. En el cuarto capítulo se realiza una exposición detallada de los pasos dados para el desarrollo del IDS, explicando además las características que se han ido introduciendo paulatinamente durante el transcurso del año académico.

4. En el quinto capítulo se valida la aplicación sometiendo a ésta a una serie de escenarios de prueba. Además se explica el entorno de trabajo desarrollado para la correcta realización de estas pruebas.

Finalmente en el sexto capítulo se ofrecen las conclusiones alcanzadas tras la realización del trabajo, así como líneas de trabajo futuras para la continuación del trabajo presentado.

2 Estado del arte

2.1 Introducción

En este capítulo se realiza un repaso del Estado del Arte en cuanto a tipos de sistemas SCADA, protocolos de comunicación empleados en la transmisión de datos a través de sistemas SCADA, amenazas de ciberseguridad y finalmente, soluciones de tipo IDS.

El desarrollo de un IDS orientado a SCADA conlleva el estudio de diversos factores. En primer lugar es necesario conocer cuál es el funcionamiento habitual de una red SCADA, qué datos suelen transmitirse y cuál es la velocidad a la que se hacen estas transmisiones. Posteriormente es necesario comprender qué tipos de amenazas existen en la actualidad [6] [7], cuáles de ellas son comunes a todo tipo de redes y finalmente delimitar aquellas que son específicas para una red SCADA.

Una vez reunidos todos los datos necesarios en cuanto a amenazas y topología habitual de la red SCADA se procede al análisis de las capacidades de un IDS, en concreto de aquellos dedicados a SCADA, con el fin de cumplir los objetivos de desarrollo marcados.

2.2 SCADA

Bajo el nombre "Supervisory Control and Data Acquisition" se encuentra una de las herramientas más utilizadas en entornos industriales o entornos bajo control de los que se desea obtener una visibilidad del estado de procesos, umbrales, valores estratégicos o decisivos para el óptimo funcionamiento y producción.

Su principal función es la de recopilar datos que provienen de PLC's (*Programmable Logic Controller)*, sensores o registros de actividad. En muchos casos sería peligroso que estos datos quedaran expuestos por una mala gestión de la seguridad de red.

Como ejemplo de infraestructuras en las que es habitual encontrar redes SCADA, son de especial preocupación aquellas en las que el resultado de los procesos realizados conlleva la producción de recursos de consumo diario entre la población, tales como la producción energética o el tratamiento de aguas para el consumo humano.

2.2.1 Sistemas SCADA destacados

Se trata de un ámbito de desarrollo con una clara tendencia hacia el software propietario, siendo pocas las herramientas SCADA de uso libre que además se complementen adecuadamente a las características del entorno de trabajo desarrollado.

Esto sistemas permiten principalmente hacer un control exhaustivo de parámetros en un entorno industrial, como pueden ser presiones, niveles, velocidades y concentraciones de diferentes áreas.

Comerciales
Algunos de los fabricantes más reconocidos del mundo ofrecen soluciones modulares para empresas en las que se suelen integrar numerosos controles. Este tipo de sistemas suponen un reto para los sistemas de detección de intrusiones, ya que en muchos casos las conexiones entrantes a controlar podrían llegar a ser inabordables.

En general, estos sistemas son utilizados en industrias como la farmacéutica, la automovilística o el tratamiento de aguas, como es el caso de Siemens System Simatic WinCC o Schneider CITECT SCADA. Encontramos algunas excepciones a la solución modular, como por ejemplo el sistema InTouch de Wonderware.

Open Source
Además de las anteriormente nombradas existen otras soluciones como OpenScada, cuyo despliegue es modular en semejanza a las opciones comerciales y entre cuyos protocolos de comunicación se encuentra ModbusTCP, RapidScada, que igualmente soporta este protocolo, o el sistema SCADA Mango Automation de Infinite Automation Systems.

2.2.2 Protocolos industriales presentes en redes SCADA

Existen diversos protocolos que permiten la transmisión de datos de control en redes industriales. Entre estos destacan DNP3, Profinet y Modbus [8]. Dada la escasa información de la que se disponía en el momento de iniciar este Trabajo de Fin de Grado, la elección de ModbusTCP permitió realizar un arranque algo más rápido en las tareas planificadas que si el protocolo elegido hubiese sido otro.

En este entorno de trabajo es habitual encontrar redes SCADA transmitiendo y recibiendo datos a través de este protocolo ModbusTCP, que trabaja sobre TCP agregando una serie de campos a continuación de la habitual cabecera de este protocolo, (que más tarde en este trabajo se detallan), para permitir la recepción y transmisión de las peticiones y respuestas, así como los distintos tipos de lecturas, escrituras u operaciones generales que permite el protocolo.

2.2.3 ModbusTCP

El protocolo ModbusTCP se basa en su predecesor de uso con puerto serie [9]. Permite realizar una comunicación a nivel aplicación entre dos o más equipos, siendo esta comunicación siempre sobre Ethernet y TCP/IP. Combina de esta manera una red física, un estándar de comunicación de red y un método de representación estándar a nivel de aplicación [10]. Su uso se basa en la definición de una serie de funciones que permiten realizar escrituras, lecturas, paradas o interrupciones del servicio.

En el ámbito de este Trabajo de Fin de Grado se va a considerar siempre que la comunicación típica conlleva una mayoría o incluso una totalidad de paquetes con solicitudes de lectura, siendo en menor medida usuales las solicitudes de escritura y tomando como posibles amenazas todas aquellas funciones distintas de estas dos primeras.

En este caso, TCP/IP es un mero protocolo de transporte que se ofrece como base para el protocolo de aplicación, el cual define el significado de los datos y su estructura. La comunicación típica en este caso es siempre de un mismo estilo independiente de la función seleccionada. La tasa habitual que se estudia en este Trabajo de Fin de Grado es de una petición por segundo, significando esto que cada petición supone un tráfico de tres paquetes por segundo y siendo esta tasa directamente proporcional al número de clientes conectados y realizando peticiones al servidor del sistema SCADA.

El principal problema del uso de ModbusTCP es que los datos transportados no están cifrados, lo que permite una perfecta visualización de éstos con una simple captura del tráfico de la red y su posterior análisis con herramientas como Wireshark.

Otro de los graves problemas de ModbusTCP es la herencia tecnológica que recibe de anteriores versiones: su creación data de finales de los años 70 y en ese momento la variante TCP no entró en los planes de diseño. La situación normal de funcionamiento de uno de estos sistemas era mediante la interconexión cableada de los equipos, controladores y sensores, de manera que no resultaba necesaria la codificación de estos datos. También, es importante aclarar que la variante ModbusTCP no se ideó con la intención de hacer posible la conexión remota, siendo el despliegue tipo uno en el que los equipos funcionaban en un entorno de red local controlada y sin acceso al exterior de ésta.

El siguiente paso lógico es permitir la comunicación remota de los equipos, siendo posible controlarlos desde salas de gestión de incidencias o centros de procesamiento de datos. Una vez llegados a este punto bastaría conocer la IP del servidor de datos y el puerto por el que habitualmente recibe peticiones un sistema SCADA sobre ModbusTCP para comenzar a inyectar o recibir datos, obtener un patrón de funcionamiento, observar qué puertos y qué funciones aparecen habitualmente.

2.3 Seguridad

2.3.1 Situación actual

En este apartado se estudian las principales amenazas relativas a los sistemas de detección de intrusiones: barridos a gran escala, ataques de denegación de servicio y explotación de vulnerabilidades [6]. Cada uno de estos ataques tiene un propósito bien definido que proporciona a los atacantes un beneficio relativo al esfuerzo realizado y a la complejidad del ataque.

Barridos a gran escala

El principal objetivo de un barrido de puertos es el de recopilar información de interés para un posible ataque futuro. Habitualmente estos barridos concentran sus esfuerzos en localizar puertos que se encuentren desprotegidos para una determinada dirección IP.

Estos barridos buscan ocultar su rastro y dificultar que los sistemas de detección puedan alertar de la incidencia mediante técnicas de aleatoriedad en el barrido de puertos, aplicándose esta aleatoriedad al orden de búsqueda de puertos vulnerables o a la frecuencia con la que se realiza el escaneo.

Un ejemplo de esto puede ser testado realizando ráfagas variables de paquetes: un mayor número primero y un número menor más tarde, o aumentar la tasa aleatoriamente para ocultar paquetes sensibles de ser detectados y catalogados como una alerta.

Igualmente, el barrido de puertos puede realizarse teniendo en cuenta la topología de la red atacada. Si, como en el caso de estudio de este Trabajo de Fin de Grado, la aplicación susceptible de ser atacada concentra su transmisión en un único puerto, el atacante podría optar por realizar un barrido de un único puerto en diversas IP's con el fin de localizar los servidores de datos.

En el caso contrario, se puede realizar un ataque que concentre sus esfuerzos en analizar el mayor número de puertos posible aunque el resultado que ofrece sea más difuso en la información recabada. En concreto, una de las herramientas más utilizadas en el mundo de la seguridad en entornos IT es Nmap, que permite realizar ataques sencillos y rápidos a dominios, IP's e incluso rangos de estas últimas sin tener conocimientos excesivamente avanzados sobre la materia.

Virus

El comúnmente llamado virus, introducido en un sistema como el que concierne a este desarrollo, puede provocar el funcionamiento erróneo de los sistemas o la parada completa de los mismos. Cierto es que para activar su mecanismo es necesario que se produzca un error humano, un engaño por parte del atacante en el que se consiga introducir el "gusano" en el sistema o bien que un "insider" sea el que introduzca él mismo el código malicioso.

Uno de los efectos colaterales de la activación del mecanismo del virus es su posible autorreplicación enviándose a los dispositivos interconectados con el sistema, de manera que provoque una sobrecarga en la red tanto por su intención de expandirse como por su segura intención de reducir el rendimiento del sistema.

Dado que en la actualidad la tendencia y motivación de estos ataques se inclina hacia el beneficio económico, es posible que uno de estos virus no busque tanto la denegación del servicio como la obtención de datos sensibles desde dentro del propio sistema.

Particularizando en el sistema que concierne a este Trabajo de Fin de Grado, un virus no es objeto de estudio dado que el IDS desarrollado centra sus esfuerzos en la observación y análisis del tráfico de red, por lo que únicamente se podrían observar los efectos posteriores a su activación, suponiendo siempre que la intención de este virus sea la de detener el envío de información o provocar un aumento o reducción de la tasa de transmisión de forma drástica.

Denegación de servicio distribuida

Un ataque DDOS (*Distributed Denial of Service)* permite al atacante interrumpir el servicio ya sea por completo o de facto (funcionamiento extremadamente lento). Este tipo de ataques basan su poder en la multiplicidad de atacantes. Para ello se infectan una serie de dispositivos de tal manera que estos realizaran la labor de "amplificadores" del ataque.

Típicamente se observan dos estructuras de ataque DDOS: inundación de paquetes SYN y ataque mediante el uso de reflectores o emisores de paquetes.

En el primer caso, la potencia del ataque se basa en la construcción de múltiples paquetes con una IP de origen falsa, de manera que el habitual establecimiento de comunicación TCP queda a medio camino. Una única petición de este tipo no afecta en gran medida, pero cuando se multiplica por miles el número de peticiones y éstas se concentran en un único servidor, el efecto inmediato es el de bloqueo de la red si no se gestiona correctamente este tráfico.

En el segundo caso, el poder del ataque se basa en la retransmisión de paquetes a través de servidores DNS, routers o servidores web. Estos paquetes se transmiten a los intermediarios en lugar de dirigirlos directamente al objetivo, pero con la dirección IP del objetivo del ataque como IP origen del paquete. De esta manera se provoca que el objetivo del ataque se inunde con respuestas a paquetes SYN que él no ha enviado y que no espera recibir (SYN-ACK). Este procedimiento es el encargado de generar el denominado "spoofed traffic". A efectos prácticos los dos casos se estudiarán como uno único en lo que concierne al desarrollo del IDS para ModbusTCP.

2.3.2 Seguridad en redes SCADA

Los principales eventos susceptibles de convertirse en potenciales amenazas en entornos SCADA son aquellos que conllevan el establecimiento de una nueva conexión. Dado que esta conexión puede ser perfectamente legítima, el umbral diferencial que permita identificar una posible amenaza debe ser estudiado con especial atención. En el caso de este TFG, una nueva conexión siempre supondrá un aviso del IDS, ya sea en forma de un aumento en la tasa de transmisión o a través de la detección de determinadas banderas de señalización.

Otros de los eventos señalados que merecen especial atención son aquellos que suponen un cambio en la configuración o en el modo de funcionamiento del sistema, como sería un cambio en la tasa, en el número de datos a transmitir o comandos que permitan realizar paradas en el servicio, reinicios o transmisión de la configuración de transmisión actual.

2.3.3 Sistemas IDS

Un IDS (*Intrusion Detection System*) o Sistema de Detección de Intrusiones es un sistema que, mediante análisis en tiempo real del tráfico de red, es capaz de discernir entre tráfico legítimo e ilegítimo en base a una serie de condiciones que pueden ser más o menos específicas en función de la red en la que se integre el sistema.

Habitualmente un IDS se despliega en aquellos puntos de la red en los que se producen conexiones entrantes que realizan solicitudes a servidores. En todo caso, un IDS únicamente detecta conexiones como posibles amenazas sin llegar a bloquear o descartar el tráfico entrante.

Ocasionalmente, y en función de las necesidades de la red, un IDS puede estar integrado en un IPS o Sistema de Prevención de Intrusiones (*Intrusion Prevention System*), que se suele desplegar en aquellos puntos que exijan una actuación rápida en caso de amenaza. Un IPS demanda muchos más recursos y es por ello que habitualmente suelen ser dispositivos hardware con procesadores de propósito específico de red.

Dependiendo del alcance de funcionamiento del IDS diseñado, requerirá de unas características hardware determinadas y de un tiempo de "aprendizaje" que permita descartar de antemano posibles falsos positivos que oculten o apantallen aquellas verdaderas amenazas de la red.

En la mayoría de los casos, un IDS/IPS se integra en algún equipo cortafuegos desplegado en la salida a internet de una red determinada, o en zonas de la red que reciban peticiones generalizadas del resto de la red, sin discriminación de origen. Algunos ejemplos de este tipo de arquitectura son los firewall de nueva generación de empresas como Fortinet o Palo Alto Networks.

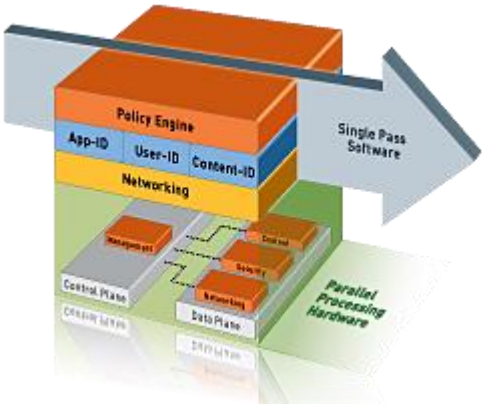


Figura 2-1: Esquema de funcionamiento de un firewall Palo Alto

En otros casos, sobretodo en entornos en los que se necesita de una mayor capacidad de los módulos IDS/IPS, se despliega una unidad independiente del resto del firewall que realiza las tareas de detección y prevención. Dichos equipos cuentan con hardware y arquitecturas dedicadas para la gestión de tráfico y análisis del mismo. Como ejemplo de la arquitectura descrita se puede destacar los equipos de la empresa Corero.

Figura 2-2: IPS de la empresa Corero

2.3.4 IDS para redes SCADA

Para el desarrollo de este IDS interesa conocer algunas de las características principales de los sistemas ya desarrollados entorno a algunos sistemas SCADA. En concreto, en el siguiente apartado se describen las características de tres diferentes tipos de IDS: IDS con detección por anomalía, por firma y *Stateful Protocol Analysis* IDS [4].

Detección por firma

Una firma es un patrón o cadena identificable que se relaciona de manera directa con un ataque o conexión ilegítima. Por tanto, un IDS con capacidad de detección por firma será capaz de analizar el tráfico de red en busca de estos patrones.

Normalmente estos patrones ya quedan predefinidos para que el IDS trabaje en base a una situación ya conocida. Para patrones desconocidos, el IDS que se desarrolle según este concepto deberá apoyarse en otros sistemas para reconocer nuevos patrones o utilizar técnicas de aprendizaje máquina que permitan la generación de estos patrones en un tiempo reducido. Esta detección es efectiva contra ataques ya conocidos.

Análisis de protocolo por estado (SPA)

Ese tipo de detección se centra en el análisis del estado del protocolo, es decir, es capaz de predecir si se está siguiendo el curso habitual de la comunicación y lanza una alerta si de alguna manera este proceso se altera o interrumpe. Esencialmente puede parecerse mucho a un IDS basado en anomalía, pero difieren en la forma en la que tratan al tráfico.

Su naturaleza exige un código más complejo que sea capaz de analizar en todo momento tanto su propio estado, lo que implica almacenar temporalmente la situación en la que se encuentra la ejecución del código del sistema de detección de intrusiones, como los paquetes recibidos en la transmisión.

Detección por anomalía

Una anomalía es una desviación sobre un comportamiento esperado. Este enfoque permite generar estadísticas desde cero y asociar éstas a comportamientos esperados en los sistemas.

Las estadísticas pueden ser estáticas, lo que exige detener en algún momento la toma de datos para esta generación, o dinámicas, permitiendo mantener un IDS adaptativo a las condiciones que se den en cada instante de tiempo reduciendo ligeramente su capacidad instantánea de detección. Este método de detección tiende a ser utilizado cuando las amenazas son desconocidas.

El IDS desarrollado en este Trabajo de Fin de Grado parte de la base de una detección por anomalía, extendiendo este tipo de detección no sólo a la anomalía referida a la tasa de transmisión de paquetes, sino de manera similar a las anomalías producidas en determinadas características esperadas a los paquetes [11]. Esto es posible realizarlo por las especiales características de la red con la que se trabaja durante este desarrollo.

Además, con el fin de perfeccionar el análisis y la de detección así como evitar posibles errores en la señalización de alarmas se amplían los objetivos a alcanzar en el desarrollo, tal y como se detalla en el siguiente capítulo. Se busca principalmente caracterizar la comunicación estándar entre servidor SCADA y cliente identificando los detalles y posibles bits diferenciadores entre paquetes cuando se produce un ataque a la red.

Esta aproximación logra señalizar amenazas que no supongan variaciones anómalas del flujo y que sin embargo provoquen cambios en la estructura y/o tamaño de los paquetes. Este tipo de búsqueda se realiza en todo paquete recibido sin distinción del protocolo utilizado para una determinada transmisión

2.4 Conclusiones

Algunos autores han generado nuevas clasificaciones que subdividen y reorganizan los métodos de detección en función de la procedencia de los datos y los tipos de amenazas que detectan, pero a efectos prácticos de éste desarrollo, la clasificación que servirá de base será la anteriormente comentada.

El objetivo de este Trabajo de Fin de Grado es, por tanto, desarrollar un IDS en base a los conocimientos encontrados en esta clasificación, en concreto siguiendo las directrices básicas de la detección por anomalía. El IDS desarrollado tiene por objetivo analizar tráfico específico del protocolo ModbusTCP y detectar posibles amenazas o desviaciones del flujo normal de tráfico presente en un sistema SCADA.

De manera adicional, se toman ideas provenientes de la detección SPA para adaptarlas a un modelo de detección por anomalía. La finalidad es generar una serie de estadísticas auxiliares que permitan discernir [5] el nivel de amenaza al que se enfrenta la red sin dejar por ello de prestar atención al objetivo inicial de este Trabajo de Fin de Grado.

Finalmente, el sistema de detección de intrusiones o IDS se somete a una serie de pruebas que validan su comportamiento frente a amenazas y su capacidad de detección en tiempo real, indispensable requisito de cualquier IDS en la actualidad.

En el siguiente capítulo se determinan los requisitos funcionales y no funcionales que se establecen como objetivos a desarrollar y que aparecen más tarde en el cuarto capítulo.

3 Análisis de requisitos

3.1 Introducción

Sentadas las bases de conocimiento sobre las que desarrollar un sistema de detección de intrusiones, la siguiente etapa en este trabajo es la de establecer las funcionalidades y requisitos deseados para el sistema. Dada la naturaleza crítica de la detección temprana es necesario que el funcionamiento del sistema desarrollado cumpla con las características de un análisis del tráfico en tiempo real.

En este capítulo se describen los principales requisitos funcionales que debe cumplir el IDS desarrollado, como la detección de variaciones anómalas del flujo de tráfico recibido a través de ciertos puertos, y los requisitos no funcionales, referentes al entorno en el que se desarrolla el IDS o su rendimiento frente a determinadas pruebas.

3.2 Requisitos funcionales

Como se menciona en el apartado anterior, el principal requisito es la detección de flujos anómalos de tráfico en tiempo real. Es necesario que el sistema sea capaz de alertar de la conexión de nuevos clientes, aumentos en la tasa de transmisión y/o conexiones no legítimas.

Adicionalmente, se requiere la detección de tamaños anómalos en los paquetes, ya que éstos cumplen generalmente un tamaño determinado. Esto incluye y se complementa con la detección de fragmentación de paquetes y permite percibir cualquier tipo de comunicación que mediante fuerza bruta trate de interrumpir la transmisión canónica de ModbusTCP.

Con el fin de afinar en mayor medida la detección de posibles amenazas, se requiere la detección de banderas TCP no presentes en la estructura habitual de la comunicación ModbusTCP. Estas banderas permiten la detección de posibles fallos estructurales de la comunicación o desviaciones en la naturaleza de los paquetes recibidos. En algunos casos es de esperar que estas estadísticas disparen alertas de manera simultánea a las detecciones por anomalía temporal o de tamaño de paquetes.

Otro de los requisitos será la detección de funciones ModbusTCP no habituales en el flujo típico de una comunicación entre servidor y cliente. En el caso del sistema desarrollado, se han tomado como habituales las funciones relacionados con la lectura, tanto en bits de señalización como en registros mayores a un bit. Como función no tan habitual pero legítima se ha considerado la escritura en estos registros y bits de señalización. Adicionalmente, y dada la naturaleza y origen del protocolo ModbusTCP, se han considerado otras funciones como no típicas y por tanto susceptibles de constituir una amenaza para la red. En esta categoría recaen aquellas que buscan conocer datos del servidor, órdenes de reinicio o parada y códigos de función no reconocidos, como aquellos pertenecientes a la variante serie del protocolo.

Todos estos sistemas de detección deberán a su vez alertar de cualquier anomalía y con el fin de establecer una clasificación y nivel de amenaza, se les otorga una ponderación previa a la realización de las pruebas que además sugiera cómo de eficiente en la detección es el sistema desarrollado.

3.3 Requisitos no funcionales

En este apartado se describen los requisitos necesarios para que las pruebas de validación necesarias sean realmente adecuadas.

Para ello, en primer lugar se debe crear un entorno de pruebas fiable y que no introduzca distorsiones a la comunicación entre servidor y clientes. La solución deseada consiste en un número determinado de máquinas virtuales, de diferente propósito y diferente sistema operativo en algunos casos, pero interconectadas mediante una red local que sirva de base para la comunicación a través del protocolo ModbusTCP.

Estas máquinas virtuales deben reunir los requisitos necesarios para un análisis exhaustivo del tráfico y de igual manera no afectar al rendimiento tanto de la transmisión de los paquetes y datos habituales del protocolo ModbusTCP. Se estudian algunas distribuciones Linux que a priori se consideran aptas para este tipo de estudios y algunas que por facilidad de desplieguen permiten avanzar con rapidez inicialmente.

De igual manera se establecen unos requisitos óptimos que otorgan a cada una de estas máquinas la misma funcionalidad a efectos prácticos que la de un ordenador común. Estos requisitos son inicialmente de 2 Gb de memoria RAM y 25-30 GB de memoria de almacenamiento, empleando 2 procesadores del ordenador anfitrión y 2 núcleos por procesador. La gestión de recursos en esta última área depende totalmente del sistema operativo anfitrión, que en este caso es Windows 10 en el momento de la realización de las pruebas.

En lo referido al rendimiento, la necesaria funcionalidad en tiempo real ajusta ése a los tiempos habituales de transmisión del protocolo y a las tasas de transmisión alcanzadas en las pruebas iniciales con el protocolo ModbusTCP que se comentarán más adelante. Dentro de estos tiempos, se considerará aceptable que el sistema sea capaz de tomar los paquetes dirigidos al servidor (que normalmente recibirá peticiones a través de un puerto determinado) y que no pierda los sucesivos, siempre y cuando se permita el análisis en tiempo real de manera continua del tráfico entrante.

3.4 Conclusiones

Mediante el establecimiento de estos requisitos es posible comenzar a describir y diseñar todas las opciones que este desarrollo presenta. De manera esencial es necesario que, para el funcionamiento mínimo aceptable de la solución, toda variación anómala en el flujo de llegada de paquetes sea alertada, así como cualquier variación en el tamaño de paquetes que se salga del margen habitual de bytes que se da en la comunicación entre un servidor y un cliente ModbusTCP. En el siguiente capítulo se describe paso a paso el proceso de desarrollo de este IDS.

4 Desarrollo de la solución

4.1 Introducción

Durante el cuarto capítulo se desglosa y detalla la solución ofrecida para el análisis y detección de tráfico de origen no reconocido en el entorno de un sistema SCADA. Se explica en primer lugar la fase de detección del tráfico en tiempo real y posteriormente su análisis en la segunda fase. La siguiente figura detalla el comportamiento del IDS frente al tráfico recibido.

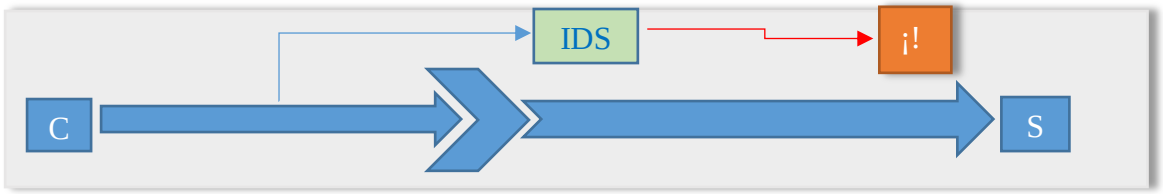


Figura 4-1: Esquema de avance del tráfico de red

Cómo se observa, el tráfico fluye sin parada ya que un IDS que analiza tráfico de tipo TCP no debe interferir en la transmisión de modo alguno que no sea el de monitorizar y alertar de posibles amenazas. (Si este tráfico fuese de otro tipo: HTTP, SMTP, etc. si se podría establecer que el IDS realizase un análisis del tráfico a posteriori y no permitiese el flujo de paquetes hasta terminar el análisis).

4.2 Captura del tráfico

4.2.1 Recepción de paquetes

La primera fase del IDS implica la recepción del tráfico mediante una serie de funciones predefinidas en las bibliotecas elegidas. Para la recepción de tráfico es necesario realizar una serie de pasos previos que se describen a continuación. Además, se estudia cómo se almacenan momentáneamente los datos recibidos para su análisis posterior.

En las próximas páginas se detalla el desarrollo del “sniffer” y su puesta en marcha junto a los pasos previos a realizar antes de proceder con la operación de escucha de tráfico. Para el desarrollo de este “sniffer” se opta por el lenguaje de programación Python, tanto por sencillez de uso como por las librerías disponibles que se comentan a continuación.

Este “sniffer” requiere de una serie de librerías externas que se agregan a Python, como “pcap”, que permite la captura de paquetes en tiempo real o “numexpr”, que permite realizar un cómputo optimizado de los cálculos a fin de no retrasar demasiado cada paso del desarrollo.

En la siguiente figura se muestra la relación completa de librerías usadas o que se necesitan por dependencia de otras. De algunas librerías sólo ha sido necesario importar una sección.

```
from __future__ import division
import pcap
import ipaddress
import string
import struct
import time
import binascii
import sys
import numpy as NP
import numexpr as NE
from binascii import hexlify
```

Figura 4-2: Librerías importadas

Se entiende que en un entorno de producción, el cual requiere de un funcionamiento continuo de la solución, estos pasos previos únicamente se llevarían a cabo en la puesta en marcha inicial o en posibles reinicios/caídas del sistema. La posible disminución del rendimiento o el tiempo de convergencia hacia el comportamiento óptimo de la solución quedarían minimizados, bien por ser la anteriormente mencionada puesta en producción, bien por ser un entorno redundado ante la más que probable criticidad del análisis de amenazas.

En primer lugar, el sistema requiere de una interfaz por la cual observar el tráfico en ambas direcciones. La interfaz se establece en el momento de la ejecución del script. En el caso de este desarrollo la interfaz es "eth0", que simula (por su situación dentro de una máquina virtual) ser una interfaz Ethernet cableada.

```
root@sec:~/Escritorio/IDS/TFG_Pruebas# sudo python2 ServerIDS093_Rev.py eth0
```

Figura 4-3: Comando de ejecución del IDS

En segundo lugar, se proporciona un filtro que permite observar el tráfico en el puerto de funcionamiento de SCADA (502). Se considera que todo puerto distinto del 502 queda cerrado y no apto para la transmisión de tráfico ModbusTCP.

Este paso queda definido a través de la instrucción que se observa en la imagen:

```
filter="ip and host 192.168.41.141 and port 502"
```

Figura 4-4: Filtro Configurable

Mediante este filtro también se configura la dirección IP en la cual se pretende escuchar y observar el flujo de tráfico ModbusTCP. Los comandos a usar son similares a los usados para filtrar paquetes en Wireshark. Una vez creado este filtro, es posible iniciar la sesión de escucha mediante la línea:

```
h=p.open_live(dev,snaplen, promisc, msec)
```

Figura 4-5: Apertura de sesión de *sniffing*

Las opciones incluidas en esta función no son decisivas y se recomienda dejar tal cual aparecen en algunas pruebas de concepto referentes a la librería pcap.

Se pasa este filtro a la siguiente función, la cual se encarga de establecer e indicar la máscara que previamente se habrá detectado mediante la función "pcap.lookupdev":

```
p.setfilter(filter,0,mask)
```

Figura 4-6: Establecimiento del filtro

Tras estos pasos, es necesario establecer una sesión de control mediante un sniffer, que da paso a un bucle con un determinado tiempo de duración inferior o igual a 1 segundo, dentro del cual se hace una llamada a la función que se encarga de procesar los paquetes para que sean legibles. Esta función, denominada "process_pkt" se define de tal manera que es capaz de recoger la información referente a la longitud de los paquetes, sus datos y el tiempo de llegada del paquete.

4.2.2 Almacenamiento y contabilidad de paquetes

Con los datos llegando de manera continuada, es necesario establecer una serie de variables que sean las encargadas de almacenar momentáneamente la información para que ésta sea tratada correctamente y se pueda establecer si el tráfico recibido es o no parte de un ataque o amenaza para la red.

Esencialmente se almacenan los datos, mediante una lista denominada "data_buff", en la cual cada posición corresponde a un conjunto de bytes recibidos por paquete. De esta manera, el análisis posterior conlleva únicamente conocer la posición del paquete en la lista y dentro de éste, la posición de los bytes que son objeto de estudio para la detección de anomalías en el flujo de tráfico. Una vez finalizada la ejecución del script, estas listas se reinician para su uso en la siguiente iteración.

```
if p.loop(1,process_pkt)== -1 :

        print "Captura errónea de paquetes"
else :
        prev_length.append(len(datos))
        data_buff.append(datos)

        pktlen_aux= NE.evaluate('pktlen_aux + pktlon')

        if Test == 1 :

                PKT_TB = PKT_TB +1      # Medida para limitar pruebas
                print ("%d PKT_TB" %PKT_TB)


        if pkt_s == 0:                          ## 1ª Iteración de la cuenta de paquetes por segundo

                initial_time=glob_timestamp     ## Se marca el inicio de la medida
                pkt_s=NE.evaluate('pkt_s+1')    ## Se suma uno al número total en este segundo.

        else :

                pkt_s=NE.evaluate('pkt_s+1')                            ## Numero de paquetes por segundo

        ## Hace que aumente el delta hasta que alcanza el segundo

                deltatime=NE.evaluate('glob_timestamp - initial_time')

time1= time.time()
```

Figura 4-7: Captura de paquetes y evaluación de intervalos de tiempo

El grado de complejidad de esta zona del código ha de ser lo más sencillo posible a fin de no retrasar el análisis en tiempo real del tráfico.

La ventana de captura de tráfico es ajustable manualmente, siendo necesario en todos los casos que sea menor a 1 segundo y siempre cerca de este valor, con el fin de mostrar datos fiables sobre las anomalías. En el siguiente capítulo se observarán algunos ajustes realizados con el fin de optimizar los tiempos de ejecución y que no afecten a la sucesión de eventos que desembocan en la detección de anomalías.

Adicionalmente, durante la fase previa de captura es necesario contabilizar el número de paquetes recibidos por segundo, tarea que realiza un simple contador al que se le añade una unidad cada vez que se recibe un paquete. La funcionabilidad interna de pcap.loop y process_pkt permiten que este contador funcione, ya que únicamente se contabiliza en el momento en el que se recibe el paquete. También se usan los llamados "timestamps", que permiten en este caso conocer cuál es el primer y último paquete recibido.

Como medida de seguridad en la toma de los datos se mantiene monitorizada la salida de la función "pcap.loop", y por consiguiente la de "process_pkt", de manera que cualquier posible error evita la suma de una unidad a la cantidad de paquetes recibidos.

4.3 Análisis de tráfico

4.3.1 Descripción del paquete tipo esperado

La principal misión de esta fase es la de observar y documentar detalladamente el comportamiento real del tráfico, así como enfrentar este flujo real con el comportamiento esperado por parte de una comunicación basada en ModbusTCP. Para ello, en primer lugar se explica la estructura interna de un paquete del protocolo ModbusTCP, así como la comunicación habitual esperada entre cliente y servidor y los principales campos que se observan y analizan para detectar posibles amenazas.

Como ya se ha descrito anteriormente, ModbusTCP es un protocolo que se transmite sobre TCP.

La estructura básica según se aprecia en Wireshark es la que se incluye en la siguiente figura:

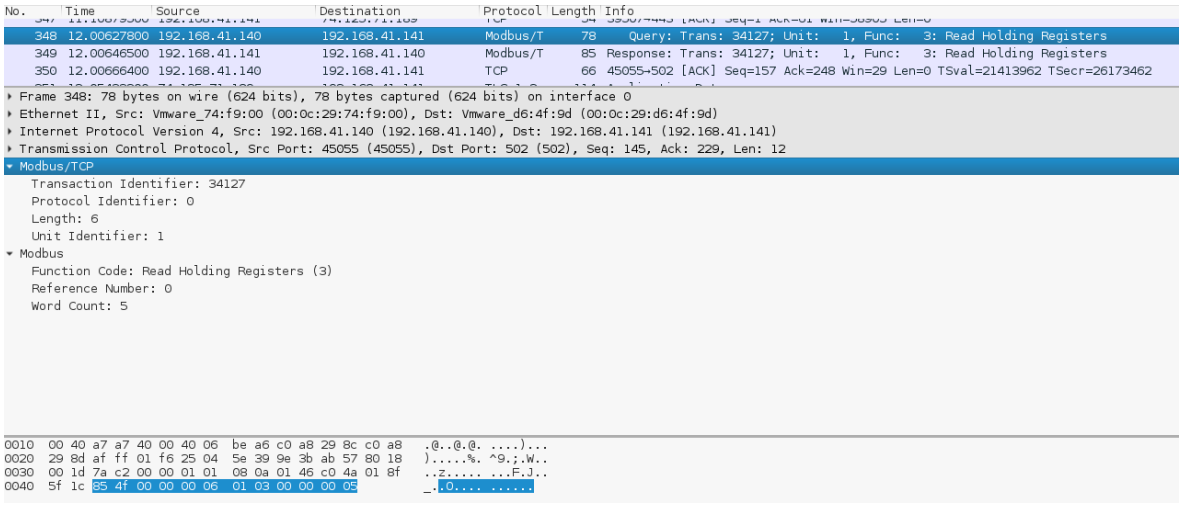


Figura 4-8: Estructura observada en Wireshark

Al habitual paquete TCP se le agrega una cabecera de aplicación, llamada *Modbus Application Protocol Header*, que transporta el identificador de transacción que permite realizar un emparejamiento correcto de peticiones y respuestas, identificador de protocolo, la longitud de los datos Modbus que se envían y un identificador de unidad, que exactamente define el servidor del que se desean obtener los datos. Esto implica que con una construcción de paquetes adecuada es posible obtener datos de cualquiera de las unidades que transmitan datos desde esa IP.

Los siguientes campos corresponden al código de función, el cual permite detectar qué función se requiere del servidor, un número de referencia y un campo "word count". Mediante estos campos se establece la comunicación de ModbusTCP que se describe a continuación.

4.3.2 Descripción de la comunicación esperada entre cliente y servidor

Para la creación de este IDS se ha tenido en cuenta que la comunicación "natural" entre cliente y servidor siempre se inicia por una solicitud realizada por el cliente. Además, el establecimiento de la conexión incluye la primera petición que realiza el cliente, por lo que en todo momento la transmisión de datos unitaria entre cliente y servidor supone un intercambio de tres paquetes.

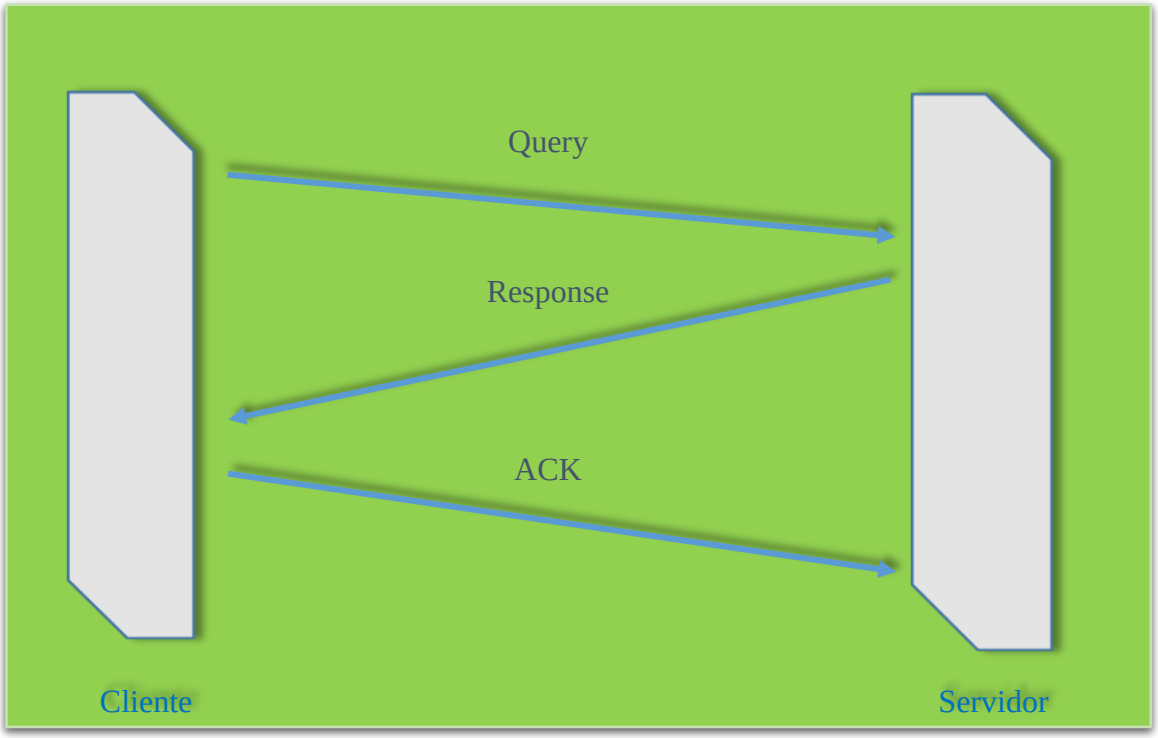


Figura 4-9: Establecimiento de conexión ModbusTCP

Cada petición del cliente únicamente requiere de ese paquete inicial de solicitud, por lo que facilita aún más la posibilidad de generar ataques con poca inteligencia puesta al servicio de la realización del mismo. El objetivo inicial es por tanto localizar cualquier posible incremento o decremento de las comunicaciones entre cliente y servidor, en números de

paquetes, que pueda indicar nuevas conexiones o intentos de conexión que busquen bloquear o dificultar el tráfico normal de la red.

Como segundo objetivo para este Trabajo de Fin de Grado se encuentra detectar posibles aumentos o disminuciones del tamaño de los paquetes, lo que denotaría un intento simple y poco costoso de realizar un ataque al sistema SCADA. Tras determinadas pruebas realizadas con el desarrollo y corroboradas estas con Wireshark se encuentra que los paquetes tienen un tamaño medio de unos 76 bytes. Dependiendo de si es la solicitud, el asentimiento a esa solicitud o la respuesta en sí misma, este tamaño varía pero se encuentra dentro de unos límites que el sistema es capaz de gestionar sin lanzar falsos positivos. Sin embargo sí que se busca que cualquier paquete no perteneciente a uno de estos tres tipos sea detectado como una amenaza.

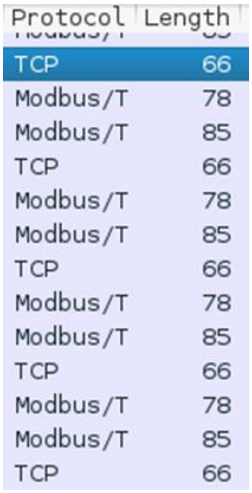


Figura 4-10: Longitud de paquetes habitual

4.3.3 Método de detección utilizado para anomalías temporales y de tamaño

El método utilizado para determinar el tamaño medio de paquetes es similar al usado para calcular la anomalía temporal.La técnica que se ha decidido utilizar, dadas las condiciones en las que se ha desarrollado el TFG y con el fin de destacar de manera clara las opciones de desarrollo y sondas de detección posibles, ha sido la técnica del suavizado exponencial.

Está técnica, evolucionada del método de promedio móvil ponderado, asume unos pesos en valor y unos valores iniciales para la media y la desviación, que en este caso permiten especificar individualmente algunos de ellos para lograr una mayor capacidad de detección. Dispone de la capacidad de autocorregirse y ajustar los pronósticos en base a las desviaciones de instantes anteriores.

De esta manera, se consigue obtener una serie de valores máximos y mínimos, relativos al número de paquetes y al tamaño medio de éstos, generando un umbral dinámico doble (superior e inferior), por el cual es posible detectar cualquier flujo de tráfico que se salga de la norma. Estas medias, desviaciones y tamaños máximos y mínimos son continuamente recalculados en base a las medias y desviaciones anteriores, así como a los valores obtenidos del cálculo directo anterior del algoritmo mediante las siguientes fórmulas:

```
PktLenDev = NE.evaluate ('Deviation_W * abs(pktlen_med - pLenMed) + (1 - Deviation_W)* PktLenDev')
pLenMed = NE.evaluate ('Media_W * pktlen_med + pLenMed * (1 - Media_W)')
```

Figura 4-11: Suavizado exponencial de media y desviación

La diferencia entre el valor instantáneo medio y el valor resultante de la anterior iteración del algoritmo quedan asociados de manera directamente proporcional al valor inicial de la desviación que se ha establecido para esa fórmula. Por otra parte, la desviación calcula para el paso anterior del algoritmo también juega un papel fundamental en el funcionamiento de este algoritmo. Es gracias a la desviación por la que es posible realizar la detección en base al tráfico real frente al esperado. Cuando se toma la media de paquetes recibidos por segundo, se observa si esta media cae dentro de los márgenes esperados por la desviación, tanto por encima del valor real como por debajo, calculando unos valores máximos y mínimos.

Con estos valores ya disponibles es posible recalcular en cada iteración los umbrales máximos y mínimos que delimitan lo que se considera un valor aceptable en el tráfico. La sección de código encargada de realizar esa tarea es la que se muestra en la siguiente figura:

```
#### Comprobación del tamaño medio de paquetes real frente al esperado ///
Comprueba la desviación en su máximo y su mínimo
    if ((pLenMed > Max_weight_value) or (pLenMed < Min_weight_value)):
        WDeviationAlarm = 1
        print ('Tamaño de paquetes desviado')
    else:
        WDeviationAlarm = 0
Definición de los valores esperados máximo y mínimo de paquetes/seg
    Max_weight_value =NE.evaluate('pLenMed + PevDown * PktLenDev')
    if NE.evaluate('pLenMed - PevDown * PktLenDev') > 0 :
        Min_weight_value = NE.evaluate('pLenMed - PevDown * PktLenDev ')
    else :
        Min_weight_value =  Min_weight_value
Tamaño medio de paquete
```

Figura 4-12: Comprobación de valores esperados

Dada la sencillez del código y su rápida implantación para cada una de las variables que se desean medir, se ha replicado de manera similar para múltiples sondas de detección que se describen a continuación.

4.3.4 Funciones de detección adicionales

El objetivo secundario de este desarrollo es el de dotar al sistema de la capacidad de identificar el nivel de amenaza en base a una serie de indicadores secundarios, capaces de detectar si el flujo al nivel de señalización y al de aplicación cumple con los valores esperados en todo momento. Como se ha comentado anteriormente, estos valores esperados se calculan mediante observación del intercambio de paquetes en Wireshark y se realiza una valoración empírica partiendo de los datos obtenidos de la repetida ejecución del script en diversas situaciones.

En primer lugar es necesario realizar un análisis exhaustivo de cada paquete recibido, siendo de especial importancia todos aquellos bytes relacionados con las TCP flags y los campos asociados a la cabecera y a las funciones ModbusTCP.

Cada uno de los paquetes observados debe pasar por todas las condiciones de manera que se determina, para cada segundo acontecido durante la ejecución del script, una cuenta que permite realizar un cálculo estimado del nivel de amenaza recibido. Cada una de estas sondas permite además que, ante un crecimiento del flujo de paquetes por segundo, estas mediciones no se vean "ensombrecidas" y que la variedad de posibles opciones permita una detección temprana de las amenazas.

```
j=0
k= 0
for i in prev_length :

    # print prev_length[k]
    buff= data_buff[k]
    ## Comprobación protocolo IP

    if hexlify(buff[12:14]) != "0800" :

        NO_IP_Count = NE.evaluate('NO_IP_Count + 1')        ## paquetes no IP
        print "Aviso: paquete no IP"
    else :

        IP_Count= NE.evaluate('IP_Count + 1')

    ## Comprobación protocolo TCP

        if hexlify(buff[23]) != "06" :

            NO_TCP_Count = NE.evaluate('NO_TCP_Count + 1')
            print "Aviso: paquete no TCP"

        else :

            if prev_length[k] <= 77 :

                NO_MDBTCP_Count = NE.evaluate('NO_MDBTCP_Count + 1')
                TCP_Count = NE.evaluate('TCP_Count + 1')
                # print ("TCP_Count : %f " %TCP_Count)
            else :
                ## print(hexlify(buff[68:70]))

                if (hexlify(buff[68:70])== "0000" and hexlify(buff[72])=="01")

                    MDBTCP_Count = NE.evaluate('MDBTCP_Count + 1')
                    # print ("MDBTCP_Count : %f " %MDBTCP_Count)
                    ##   COMPROBAR LONGITUD MODBUS  : Length_MDB(buff)

                    if prev_length[k] != len(buff) :    ## O comparar con tamañ

                        ILLEGAL_LEN = NE.evaluate('ILLEGAL_LEN + 1')
```

Figura 4-13: Comprobación de condiciones en paquetes

A continuación se pasa a describir cada una de las sondas de detección diseñadas, el motivo de su elección y el valor esperado para cada sonda en el entorno de comunicación ModbusTCP habitual. Estas sondas pertenecen a niveles variados, siendo posible distinguir entre paquetes IP, TCP/IP, ModbusTCP y determinadas funciones que veremos a continuación.

Ratio comparativo entre paquete NoIP/IP:

La misión de esta sonda es detectar cualquier flujo de paquetes que no sean de tipo IP (en este caso se soporta IPv4). De esta manera, cualquier paquete malformado o que no sea definido conforme al estándar IPv4 será detectado como una posible amenaza y se contabilizará de tal manera que el algoritmo de suavizado exponencial se encargará de gestionar para la determinación de una posible amenaza. Los bytes 12 y 13 del paquete deben ser equivalentes al hexadecimal "0800". Si esto no es así, se señala la presencia de un paquete no-IP.

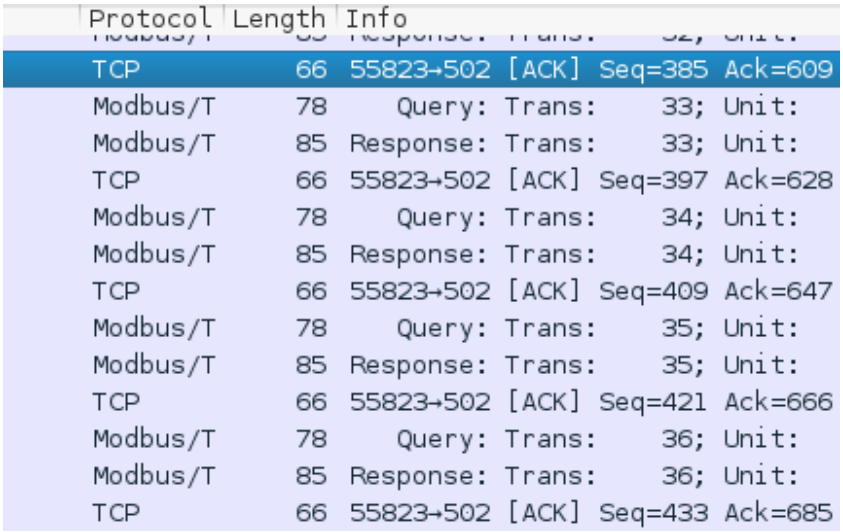
El valor esperado en estos casos para la variable "NoIP_ratio" es de 0, ya que se espera que todos los paquetes que traten de realizar un intento de conexión con el servidor sean de tipo IP y cumplan con la estructura esperada. Un único paquete de este tipo debe provocar el lanzamiento de un aviso por presencia escasa o nula en este tipo de red.

Ratio comparativo entre paquete NoTCP/TCP:

La misión de esta característica es controlar el byte 23 de cada paquete recibido. La sonda es similar en forma y propósito a la anterior, siendo extremadamente raro encontrar un paquete que no sea TCP y con petición dirigida al servidor SCADA. El valor esperado es igualmente 0 por razones equivalentes al ratio comparativo anterior.

Ratio comparativo entre paquete TCP/ModbusTCP:

El siguiente indicador o sonda difiere ligeramente en su concepto a los dos anteriores. En este caso se sabe que la red contiene paquetes de ambos tipos, siendo posible diferenciar entre los paquetes que contienen la información referente a ModbusTCP y aquellos que acompañan la comunicación para que ésta transcurra sin problema alguno y ambos extremos conozcan de la situación de la transmisión.



Protocol	Length	Info
TCP	66	55823→502 [ACK] Seq=385 Ack=609
Modbus/T	78	Query: Trans: 33; Unit:
Modbus/T	85	Response: Trans: 33; Unit:
TCP	66	55823→502 [ACK] Seq=397 Ack=628
Modbus/T	78	Query: Trans: 34; Unit:
Modbus/T	85	Response: Trans: 34; Unit:
TCP	66	55823→502 [ACK] Seq=409 Ack=647
Modbus/T	78	Query: Trans: 35; Unit:
Modbus/T	85	Response: Trans: 35; Unit:
TCP	66	55823→502 [ACK] Seq=421 Ack=666
Modbus/T	78	Query: Trans: 36; Unit:
Modbus/T	85	Response: Trans: 36; Unit:
TCP	66	55823→502 [ACK] Seq=433 Ack=685

Figura 4-14: Intercambio habitual de paquetes

A pesar de que visualmente y por la comunicación ya conocida de ModbusTCP se pueda intuir que el valor se estabiliza alrededor de 0,33 paquetes por segundo , en realidad el sistema calcula que este valor se aproxima a TMBr = 0,5 paquetes por segundo debido a la constante llegada de paquetes durante un segundo. Por tanto, la función considera que su valor esperado es éste último y en base a él recalcula constantemente la desviación permitida

Indicador "Read_Function":

El siguiente valor es uno de los más importantes debido a su capacidad de calcular el número de lecturas por segundo que se producen como fruto de una función "read" en el protocolo ModbusTCP. Este valor goza de esa importancia por la naturaleza de la comunicación ModbusTCP.

En estado de "reposo" (idealmente se espera que esto sea así en muchos casos), la mayoría de paquetes deben suponer una lectura de mediciones tomadas por sensores o indicadores que reportan su información al servidor SCADA. El cliente es por tanto principalmente observador y no debería provocar excesivas llamadas a otras funciones.

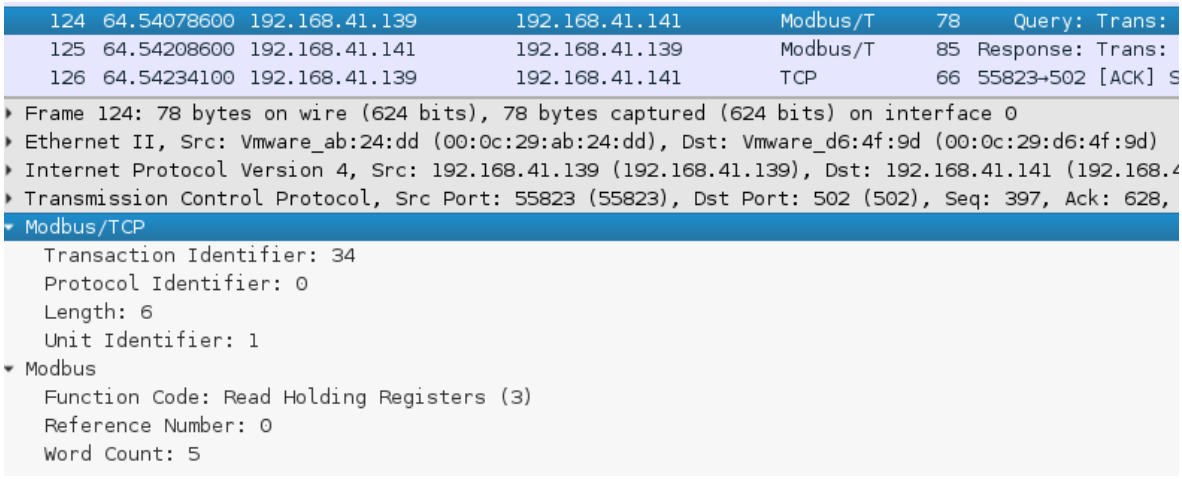


Figura 4-15: Paquete con función de lectura activa

Indicador "Write_Function":

De la misma manera que la función lectura, la función de escritura es una función menos habitual pero reconocida como legítima en el transcurso de una comunicación ModbusTCP. En el caso extremo ideal de solo lectura de este Trabajo de Fin de Grado, una escritura supone un evento de seguridad del que al menos es necesario informar y se toma éste como una posible intrusión. Por tanto el valor esperado para este índice es de 0.

Dada la similitud con el indicador anterior, es probable que, en algunos casos, un descenso en el indicador de lectura pueda ocultar un posible aumento del indicador de escritura o de alguna otra función no reconocida.

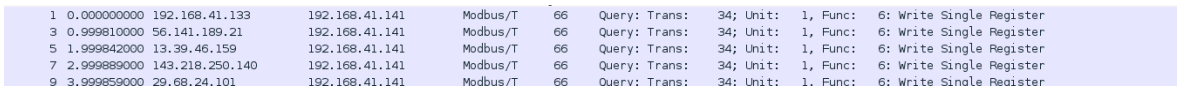


Figura 4-16: Peticiones de lectura en Wireshark

Indicador de funciones no usuales:

Si se produce una detección de cualquier función que no sea la de lectura o escritura en el contexto de una comunicación ModbusTCP, un IDS debe ser extremadamente cuidadoso y poco permisivo con las modificaciones que pueda realizar. En algunos casos, los servidores ModbusTCP permiten funciones de reinicio, parada o devuelven información esencial sobre el sistema que permiten interrumpir o limitar el correcto desempeño del sistema.

Es por esta cuestión anterior por la que es necesario monitorizar cualquier otra función como no usual, haciendo que se incremente el número de posibilidades de detectar el paquete que contenga la orden de parada, reinicio o devolución de información.

Indicador de fragmentación de paquetes:

Dado el tamaño de los paquetes de ModbusTCP, una fragmentación de paquetes no tiene sentido en el contexto de una comunicación habitual entre servidor y cliente ModbusTCP. En cualquier caso denota bien un problema en la red, bien una posible amenaza que trate de dificultar o bien interrumpir un servidor forzándole a recibir paquetes incompletos.

Este indicador está esencialmente ligado al ratio comparativo de tamaño de paquetes y a la detección por anomalía temporal, por el posible aumento tanto por tamaño como de flujo de paquetes por segundo. El valor esperado para este indicador es de cero y las pruebas realizadas en un contexto normal del tráfico demuestran que así es.

Indicador PSH_ACK por segundo (PAFr):

Este valor es uno de los descriptivos e inequívocos de todos los analizados en este desarrollo. Su valor esperado nos da a entender que 2 de cada 3 paquetes envían información y además cumplen con la habitual “charla” del protocolo TCP. Es por ello que cualquier variación, ya sea en flujo temporal o en las condiciones de los paquetes, se refleja en este indicador de una manera rápida y normalmente de forma simultánea a otros indicadores ya descritos.

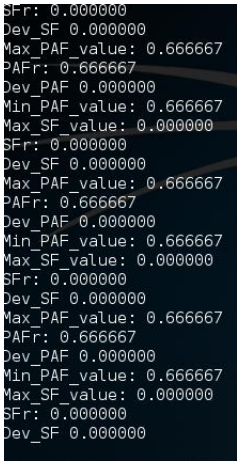


Figura 4-17: Respuesta observada de Push-Ack ratio

Indicador SYN FLAG (SFr):

El principal motivo de existencia de este indicador es el de alerta de un posible ataque de tipo SYN Flood, por el cual un atacante provoca la denegación del servicio. El valor esperado en el caso de este indicador es de cero paquetes por segundo que lo contengan, ya que en la fase de toma de datos se ha diferenciado ya entre los paquetes de establecimiento de conexión, que contienen las banderas SYN y ACK.

Es necesario destacar que, al igual que todas las demás funciones, este indicador depende de un suavizado exponencial que diferenciaría de posibles intentos de sincronización aceptables de aquellos que realmente supongan un ataque.

Indicador SYN_ACK FLAG (SAFr):

Tal y como se comenta en el indicador anterior, este valor se corresponde con los intentos de conexión que un cliente trata de realizar para la obtención de datos. Habitualmente este valor será cero, exceptuando aquellos casos en los que se produzca una nueva conexión, en cuyo caso la notificación de exceso de paquetes de este tipo va acompañada de un ascenso en el número de paquetes por segundo.

Estos dos indicadores en común permiten asociar el comportamiento a una nueva conexión, que dependiendo del entorno podrá ser considerada o no válida como conexión legítima.

Indicador ACK FLAG (AFr):

Su nombre indica su sencilla misión: realiza un recuento con periodo de un segundo sobre el número de asentimientos (únicamente los paquetes que contenga esta *flag* y ninguna más) registrados en la comunicación entre cliente y servidor. Este valor es de tipo auxiliar, ya que sus funciones están cubiertas con otros indicadores que ofrecen un grado de detalle más alto sobre el intercambio de información dentro de la red.

Indicador FIN_ACK FLAG (FAFr):

Este parámetro busca señalar cualquier posible ocasión en la que un agente externo solicita el cierre de una sesión ya abierta. Esto por ejemplo podría realizarse mediante *spoofing* con el fin de interrumpir la conexión de un cliente en concreto o incluso realizar una parada masiva de los servicios si no se gestiona correctamente.

Como valores esperados de este indicador se obtiene un valor de cero, para todos aquellos eventos que no sean una interrupción legítima del tráfico. Es importante denotar que este tipo de paquetes pueden desembocar en un ataque del tipo FIN flood”

La última opción que busca mitigar este indicador es la parada manual de servicios de un usuario que tenga acceso a la red. Cualquier acción que se encamine a esta debería ser considerada una amenaza si el entorno exige una alta disponibilidad de los sistemas

Indicador UNSUSUAL FLAG (UFr):

Finalmente, en lo que a indicadores se refiere, para todas aquellas opciones que no se hayan considerado hasta ahora queda la categoría de *flags* no usuales. El valor esperado para esta categoría es siempre cero, ya que no se observa ningún tipo de tráfico con estas características en las capturas previas realizadas durante las fases iniciales de este Trabajo de Fin de Grado

4.4 Operativa en caso de amenaza

Todos estos indicadores tienen un carácter esencialmente independiente pero cualquier análisis que se precie incluye un nivel de criticidad por el cual se pueda identificar el grado de vulnerabilidad a la que se expone la red permitiendo el paso de los paquetes maliciosos.

Para ello, se decide establecer un ranking de puntuación en base a la alerta disparada. Como el principal objetivo de este Trabajo de Fin de Grado es el de detectar flujos anómalos de tráfico, se otorga la máxima puntuación de 3 puntos a la detección de anomalía temporal.

El resumen de puntuaciones totales, con capacidad de sumar hasta 15 puntos en total es el mostrado en la siguiente tabla:

Tabla 4-1: Tabla de ponderación de alertas

Tipo de indicador	Ponderación
Detección por anomalía temporal	3,5 puntos
Detección por anomalía de tamaño	2 puntos
TMBr(TCP/ModbusRatio) PAFr(PSH_ACK) RFr(Read_Function)	1.5 puntos
Resto (10)	0,5 puntos

De esta manera, una intrusión que dispare las 2 primeras amenazas supondrá una puntuación de 5,5/15 como nivel de criticidad y una que dispare las 3 primeras categorías completas supondrá una amenaza de nivel 10/15. El sumatorio de estas ponderaciones se genera en cada una de las fases por las que se analiza el código, de manera que al finalizar el bucle se obtiene un número sin necesidad de emitir 6 amenazas diferentes por pantalla y así generar un método más visual para clasificar las posibles intrusiones.

Siguiendo este esquema se puntuarán las reacciones a las pruebas realizadas en el siguiente capítulo y se definirá el grado de funcionalidad de algunas de ellas, así como el rendimiento de este desarrollo.

4.5 Reinicio de variables y fin del bucle de captura y análisis

Finalmente, el funcionamiento de este bucle depende de un refresco y reinicio de las variables auxiliares en las que se almacenan numéricamente los eventos de seguridad. Conviene recordar que no son objeto de este reinicio aquellas variables implicadas en el algoritmo de cálculo de valores medios y desviaciones, dado que es precisamente esa la característica esencial del suavizado exponencial. Si esto se eliminase, quedaría representando únicamente por el valor estimado inicial que ya no tendría valor alguno en mitad de la ejecución.

Es importante reiniciar también aquellas listas que en un momento dado puedan aumentar de tamaño de una manera notable. Es el caso de "prev_length" que en casos de aumento del flujo de paquetes debe ser capaz de almacenar todos los paquetes y su contenido completo. Si esto último no fuese así, el desarrollo podría perder funcionalidad o interrumpir su funcionamiento debido a un problema de memoria.

```
pl = 0
pktlen_aux = 0
pktlen_med = 0
deltatime= 0
pkt_s = 0

NO_IP_Count = 0
IP_Count = 0

NO_TCP_Count = 0
TCP_Count = 0

NO_MDBTCP_Count = 0
MDBTCP_Count = 0

ILLEGAL_LEN = 0
Read_Function = 0
Write_Function = 0
Unusual_Function = 0
Fragment_Flag = 0
PSH_ACK_FLAG = 0
SYN_FLAG = 0
SYN_ACK_FLAG = 0
ACK_FLAG = 0
FIN_ACK_FLAG = 0
Unusual_FLAG = 0

prev_length = []
```

Figura 4-18: Reinicio de contadores

Las últimas tareas a realizar en este apartado son la toma de tiempos para medición del rendimiento que se puede observar en el capítulo próximo.

4.6 Conclusiones

Mediante este desarrollo, se consigue un IDS capaz de distinguir amenazas en función de múltiples variables, múltiples grados de criticidad por la suma de varias alertas y un gran número de posibles combinaciones que registran datos valiosos también a nivel de uso de red. En el siguiente capítulo se comprueba que el desarrollo es plenamente funcional y que arroja los resultados esperados en cuanto a detección de amenazas y funcionamiento en tiempo real.

5 Integración, pruebas y resultados

5.1 Introducción

En este capítulo se describe cómo se ha desarrollado el entorno de trabajo que ha permitido la realización de esta solución IDS, así como las pruebas realizadas para observar su rendimiento y los resultados obtenidos en esas pruebas. El objetivo es, por tanto, determinar qué nivel de detección es capaz de alcanzar este IDS.

5.2 Entorno de pruebas

Aunque este entorno de pruebas se encuentra explicado detalladamente en el Anexo A de este Trabajo de Fin de Grado, a continuación se detalla brevemente las herramientas utilizadas y la red desplegada para la realización de las pruebas.

En primer lugar es necesario reseñar que todas las pruebas se realizan en un entorno virtual con el fin de mantener una estabilidad que permita analizar adecuadamente las pruebas y observar posibles fallos durante el desarrollo y prueba del IDS. Esto se realiza mediante VMware Workstation 11, en el cual se crean 4 máquinas virtuales de idénticas características técnicas, capacidades de almacenamiento y sistema operativo.

El sistema operativo anfitrión de VMware es Windows 10 en el momento de la realización de las pruebas finales. Para las máquinas virtuales se opta por el sistema operativo Kali Linux. Los motivos de esta elección quedan recogidos en el anexo anteriormente mencionado.

Para simular el funcionamiento del servidor y del cliente de una comunicación basada en ModbusTCP se cuenta con las siguientes herramientas:

En primer lugar, el servidor, cuya función realiza la aplicación ModbusPal.
En segundo lugar, el cliente Modpoll, cuya función es la de solicitar determinada información sobre los registros del servidor.

Además, las máquinas virtuales creadas cuentan con tres herramientas que sirven de base para la generación de pruebas que permitan comprobar el funcionamiento de este sistema de detección de intrusiones. Estas herramientas son Nmap, que habitualmente se usa en escaneos de puertos y que proporciona los medios para probar el funcionamiento de la herramienta en el caso de un barrido del sistema en busca de datos específicos del IDS, Metasploit, paquete muy completo que permite generar casi cualquier prueba en busca de vulnerabilidades de programas y que se utiliza en estos casos de pruebas a través de una serie de *plugins* ya desarrollados que están incluidos en la herramienta y, finalmente, PackETH , generador de paquetes y ráfagas de paquetes que permite generar aquellas pruebas restantes y además realizar pruebas de estrés con el fin de comprobar el límite de funcionamiento del sistema de detección de intrusiones.

5.3 Casos de prueba

5.3.1 Tráfico legítimo

En el caso de tráfico legítimo, es decir conexión o desconexión de uno de los clientes situados en las máquinas virtuales que solo emiten peticiones desde Modpoll, el cambio que se aprecia hace subir el tráfico de 0 a 3 paquetes por segundo, y de 3 a 6 paquetes por segundo en el caso de una segunda conexión. Se considera un cliente a cada nueva ejecución de la aplicación Modpoll, aunque esta ejecución se realice desde un ordenador en el cual ya se está ejecutando una instancia de esta aplicación. Esto se considera así por simplicidad a la hora de demostrar el tráfico genérico recibido.

El tráfico habitual entonces queda caracterizado por la siguiente tabla, en la cual se muestra el crecimiento lineal de la media de paquetes por segundo en función del número de clientes conectados. Este tráfico es el habitual en un entorno SCADA de tipo ModbusTCP, en el cual el número de conexiones es muy bajo comparado a otras comunicaciones TCP.

Tabla 5-1 : Comportamiento ideal de una transmisión ModbusTCP

Número de conexiones	1	2	…	N
Número de paquetes en transmisión	3 paq/s	3x2 = 6 paq/s	…	3xN

Por cada uno de estos saltos anteriores, el desarrollo IDS lanza una alerta relacionada con un flujo de paquetes sospechoso, lo que provoca un nivel de criticidad general de 3,5 sobre 15, el cual es posible ignorar por ser una conexión que conocemos. En caso de producirse la conexión simultánea de dos clientes este nivel de criticidad no se dobla, ya que el IDS detecta como una única amenaza esa doble conexión.

Idealmente, cada una de estas nuevas conexiones proviene de un cliente diferente en máquinas separadas pero, por economía de recursos, en este estudio se permiten conexiones simultáneas desde una misma máquina virtual, lo cual no afecta de manera apreciable al rendimiento individual de cada una de las instancias del cliente.

5.3.2 Tráfico no legítimo

Se genera este tráfico mediante las tres herramientas anteriormente mencionadas: NMap, Metasploit y packETH. Nmap es una herramienta cuya principal característica es la búsqueda de puertos abiertos en una determinada red o dispositivo. Metasploit por su parte, es una suite mucho más personalizable y cuyos *exploits* ya desarrollados abarcan muchas aplicaciones y entornos distintos.

Para estas dos opciones se aprovechan las funciones y comandos ya definidos que buscan vulnerabilidades en IDS o en ModbusTCP específicamente. En el caso de la tercera aplicación, packETH exige la creación manual de los paquetes, los cuales posteriormente pueden ser transmitidos de manera individual, en ráfagas con un único tipo o en ráfagas de tipo combinado. En el ámbito de este Trabajo de Fin de Grado se desarrollan paquetes que buscan activar las señalizaciones relativas a funciones ModbusTCP, en concreto la función de escritura y funciones desconocidas.

En primer lugar, las pruebas realizadas mediante NMap, que incluyen aquellas relativas a IDS genéricos, denotan que la mayoría de los ataques inducen en el IDS una reacción conjunta de varias sondas o indicadores, que permiten realizar un pequeño análisis de funcionamiento del desarrollo. Se espera, en mayor medida, alterar el equilibrio de paquetes ModbusTCP frente a los no ModbusTCP, así como activar las balizas de alteración anómala del flujo de tráfico y algunas banderas de señalización como "Read Function", "PSH ACK" o "ACK".

Amenazas generadas por NMap

A continuación se muestran algunas de las alertas generadas a consecuencia de ataques generados con el módulo de IDS/Firewall de Nmap.

nmap -v -sS 192.168.41.141 -p 502

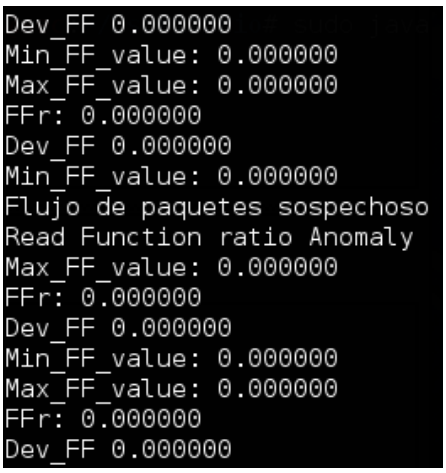


Figura 5-1: **Respuesta a nmap -v -sS**

En este primer caso se trata de imitar un escaneo mediante paquetes SYN, con objetivo la IP del servidor y el puerto 502 relativo al protocolo ModbusTCP, que suponemos conocidos por el atacante. Los resultados son los siguientes.

Como podemos observar el nivel de criticidad es bajo pero el IDS es capaz de detectar los diferentes paquetes que llegan y activan las señales de control. Se le otorga un nivel de criticidad de 3.5.

nmap -v -sA 192.168.41.141 --data-length 1000 -p 502

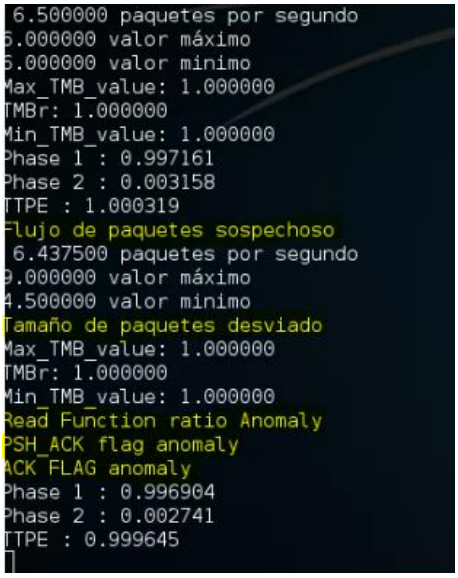


Figura 5-2: Respuesta a nmap -v -sA

A continuación se realiza una simulación de un escaneo mediante paquetes ACK, el cual como se ve, levanta mayores alertas en el sistema de detecciones de instrucciones desarrollado. En este caso el número de alertas concurrentes es de cinco. Siendo las tres últimas las que denotan con mayor precisión un cambio en el estilo de los paquetes recibidos.

Se trata de una amenaza tanto en número de paquetes, como en el ratio de paquetes de lectura o en la presencia habitual de banderas TCP. El cambio detectado en el número de paquetes cuyos bits de función ModbusTCP son anómalos, junto con la bandera de señalización ACK permiten alertar de esta incidencia. Se le otorga por tanto un nivel de criticidad 9.

Amenazas generadas por Metasploit

El siguiente paso es generar una serie de amenazas mediante la suite de Metasploit. Al ser una herramienta potente pero compleja y, aprovechando la popularidad del protocolo ModbusTCP entre algunas de las soluciones SCADA más utilizadas en la actualidad en el sector industrias, se decide tomar como válidas las diferentes opciones ya predefinidas que incluye la aplicación en el ámbito de SCADA y ModbusTCP.

En primer lugar, las opciones disponibles son las que aparecen en la siguiente figura. Se utilizan todas las opciones menos la cuarta línea, la cual únicamente despliega un cliente Modbus igual al generado con Modpoll.

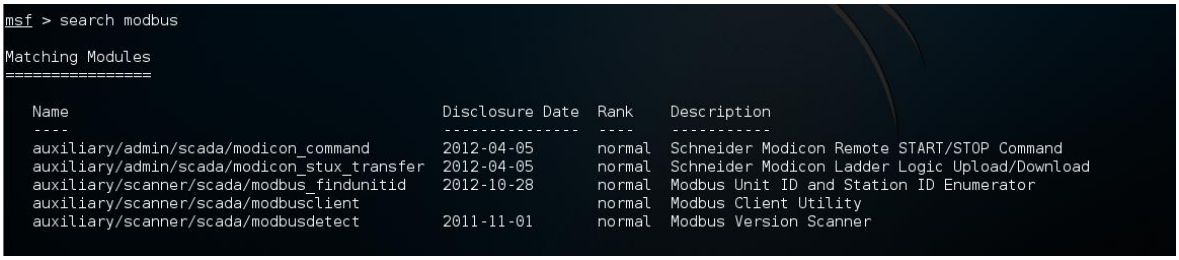


Figura 5-3: **Módulos metasploit de ataques modbus disponibles**

Las dos primeras opciones de la figura anterior, a pesar de estar pensadas para el SCADA de Schneider, cumplen con la función buscada que es alterar el comportamiento observado por el IDS y estudiar la transmisión de una posible amenaza.
Para todas las ejecuciones es necesario establecer las variables: dirección de destino (RHOST) y puerto (RPORT). En algunos casos es necesario también establecer otras opciones.

En primer lugar se ejecuta **auxiliary/admin/scada/modicon_command**, cuyos parámetros y modo de ejecución son los que aparecen en la siguiente figura.

```
msf auxiliary(modicon_command) > show options

Module options (auxiliary/admin/scada/modicon_command):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   MODE   STOP             yes       PLC command (Accepted: STOP, RUN)
   RHOST                   yes       The target address
   RPORT  502              yes       The target port

msf auxiliary(modicon_command) > set RHOST 192.168.41.141
RHOST => 192.168.41.141
msf auxiliary(modicon_command) > run
[*] Auxiliary module execution completed
```

Figura 5-4: Ejecución de modicon_command

Ante este estímulo la respuesta del IDS es la siguiente:

```
Dev_FF 0.000000
Min_FF_value: 0.000000
Max_FF_value: 0.000000
FFr: 0.000000
Dev_FF 0.000000
Min_FF_value: 0.000000
Flujo de paquetes sospechoso
Tamaño de paquetes desviado
TMB_ALARM
Read Function ratio Anomaly
Max_FF_value: 0.000000
FFr: 0.000000
Dev_FF 0.000000
Min_FF_value: 0.000000
PSH_ACK flag anomaly
SYN_FLAG anomaly
SYN_ACK flag anomaly
ACK_FLAG anomaly
Unusual Flag anomaly
Max_FF_value: 0.000000
FFr: 0.000000
Dev_FF 0.000000
Min_FF_value: 0.000000
```

Figura 5-5: Respuesta a la ejecución de modicon_command

Hasta 9 alertas son activadas al realizar esta transmisión mediante este módulo de Metasploit, siendo ésta ejecución una de las más críticas con una valor ponderado de 12.

A continuación se realiza la prueba con el segundo módulo **/auxiliary/admin/scada/modicon_stux_transfer**.

```
msf auxiliary(modicon_stux_transfer) > show options

Module options (auxiliary/admin/scada/modicon_stux_transfer):

   Name      Current Setting                                                Required  Description
   ----      ---------------                                                --------  -----------
   FILENAME  /usr/share/metasploit-framework/data/exploits/modicon_ladder.apx  yes       The file to send or rec
   MODE      SEND                                                           yes       File transfer operation
   RHOST                                                                    yes       The target address
   RPORT     502                                                            yes       The target port

msf auxiliary(modicon_stux_transfer) > set RHOST 192.168.41.141
RHOST => 192.168.41.141
msf auxiliary(modicon_stux_transfer) > run
```

Figura 5-6: Opciones configuradas para modicon_stux_transfer

La respuesta del IDS a esta ejecución es la siguiente:

```
Min_FF_value: 0.000000
Flujo de paquetes sospechoso
Tamaño de paquetes desviado
TMB_ALARM
Read Function ratio Anomaly
Max_FF_value: 0.000000
FFr: 0.000000
Dev_FF 0.000000
Min_FF_value: 0.000000
PSH_ACK flag anomaly
ACK FLAG anomaly
Max_FF_value: 0.000000
FFr: 0.000000
Dev_FF 0.000000
Min_FF_value: 0.000000
```

Figura 5-7: Respuesta a modicon_stux_transer

En este caso la ejecución provoca unos niveles de criticidad similares a los observados en las pruebas realizadas con NMAP con una nota ponderada de 10,5. Se observa cierto retraso en la aparición de varias anomalías, lo que puede denotar diferentes fases en la ejecución con el fin de apantallar alguna petición realizada por el módulo.

Para la tercera prueba, las opciones y la ejecución realizada son las siguientes:

```
msf auxiliary(modicon_stux_transfer) > use auxiliary/scanner/scada/modbus_findunitid
msf auxiliary(modbus_findunitid) > show options

Module options (auxiliary/scanner/scada/modbus_findunitid):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   BENICE        1                yes       Seconds to sleep between StationID-probes, just for beeing nice
   RHOST                          yes       The target address
   RPORT         502              yes       The target port
   TIMEOUT       2                yes       Timeout for the network probe, 0 means no timeout
   UNIT_ID_FROM  1                yes       ModBus Unit Identifier scan from value [1..254]
   UNIT_ID_TO    254              yes       ModBus Unit Identifier scan to value [UNIT_ID_FROM..254]

msf auxiliary(modbus_findunitid) > set RHOST 192.168.41.141
RHOST => 192.168.41.141
msf auxiliary(modbus_findunitid) > run
```

Figura 5-8: Opciones configurables modbus_findunitid

Los resultados ofrecidos por el IDS muestran que el ataque se desarrolla en dos fases. Una, la habitual, en la que se detecta la aparición de una serie de anomalías similares a casos anteriores (flujo de paquetes anómalos, TMB_ALARM, anomalía en PSH_ACK):

Figura 5-9: Respuesta a modbus_findunitid(1)

Y la segunda, en la que se observan anomalías en el número de SYN_ACK y FYN_ACK recibidos. Esto es posible que se deba a un descenso de estos paquetes que previamente no había sido detectado por el IDS. La buena noticia es, que gracias al carácter bidireccional de la detección en incrementos o decrementos de paquetes, se puede observar con un pequeño desfase otra serie de alertas que permiten aumentar el nivel de criticidad a 11,5.

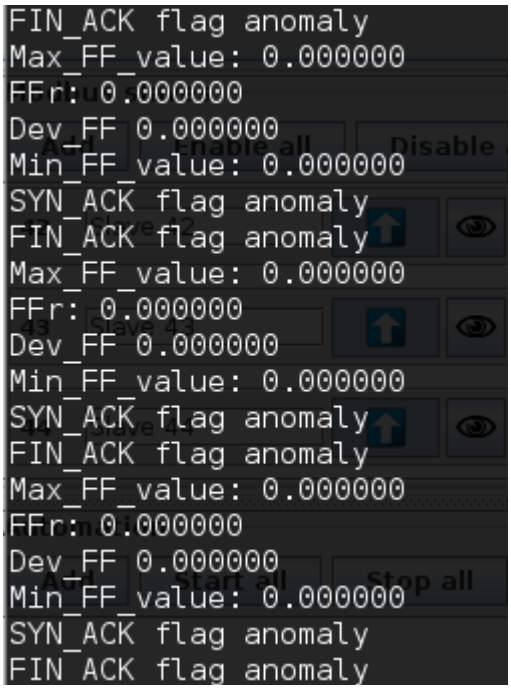


Figura 5-10 Respuesta a modbus_findunitid(2)

La cuarta prueba, por ofrecer resultados similares a la segunda, queda detallada en el anexo de pruebas.

5.4 Tasas máximas de detección de amenazas

La detección de las anomalías tiene un límite, a partir del cual se empiezan a perder paquetes en la ejecución y por tanto la fiabilidad de los resultados es menor. A pesar de esta pérdida de fiabilidad, en algunos casos la detección sigue siendo válida siempre y cuando la cantidad de paquetes que se pierden sea despreciable frente al monto total de la transmisión.

Los resultados permiten afirmar, tal y como se detalla en la siguiente tabla, que la detección es efectiva siempre y cuando la tasa de transmisión se mantenga por debajo de 1,15 Mbit/s. A partir de esta tasa se observa un drástico aumento de la pérdida de paquetes y por consiguiente de la fidelidad de los cálculos realizados para la detección de anomalías. En estos casos es posible que el cálculo obvie paquetes legítimos, aunque en el entorno habitual de un despliegue sobre ModbusTCP estas tasas son meramente experimentales y en ningún caso debería ser necesario alcanzarlas, lo cual de por sí indicaría un ataque a la red por el mero hecho de superar el umbral de la tasa de transmisión aceptable.

A continuación se incluye una tabla con la progresión y funcionamiento del IDS en los alrededores de la tasa máxima de transmisión y detección.

Tabla 5-2: Tasas de transmisión y porcentaje de paquetes recibidos.

Tasa de transmisión (kbps)	Paquetes recibidos (%)
700	100
900	100
1000	100
1100	100
1120	100
1130	100
1140	100
1145	99.66
1150	98.12
1200	97.95

5.5 Conclusiones

Tras la realización de estas pruebas queda demostrado que el IDS desarrollado es capaz de detectar algunas de las amenazas más habituales en entornos SCADA. El sistema es capaz de detectar posibles amenazas siempre que el tráfico se mantenga entre una tasa por debajo de 1,15 Mbit/s, lo cual demuestra la viabilidad del sistema para las tasas a las que habitualmente trabaja ModbusTCP. Se han realizado pruebas a diferentes tasas y longitudes de paquete en los casos en los que era posible variar estas características o bien forzarlas.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Una vez finalizadas las pruebas y observados los resultados obtenidos, queda demostrado que las técnicas utilizadas permiten construir una solución de seguridad para un sistema SCADA con las condiciones analizadas.

En primer lugar, se realiza satisfactoriamente una detección en tiempo real de cualquier amenaza, siendo necesaria una fase de un segundo de duración para alertar de una amenaza. En una red de esta naturaleza, el tiempo de detección es suficiente.

En segundo lugar, la detección de anomalías en el flujo de transmisión de paquetes funciona correctamente, lo que permite señalar con aceptable precisión cambios en la transmisión. Se añade además que esta detección cumple con los requisitos de análisis en tiempo real. Especialmente, quedan excelentemente señalizadas nuevas conexiones de clientes y variaciones abruptas del volumen de paquetes provenientes de un mismo cliente. De la misma manera, posibles escaneos de puertos y peticiones no típicas son señaladas por la detección por anomalía, aunque apoyada por otras características.

En tercer lugar, la detección por anomalía de tamaño de paquetes permite detectar satisfactoriamente cambios en la longitud de los paquetes. Cualquier posible amenaza que implique un tamaño que difiera de los márgenes habituales en paquetes ModbusTCP será detectada por esta característica del IDS.

Adicionalmente, la generación de estadísticas secundarias permite afinar adecuadamente los análisis realizados y establecer una posible clasificación en cuanto al nivel de amenaza o criticidad de la transmisión detectada. Estas estadísticas permiten en mayor o menor medida según su uso e importancia dentro de las redes SCADA observar si los paquetes recibidos conforman una comunicación adecuada.

Finalmente, gracias a todas estas características previas que se han desarrollado e implementado en la solución de detección de intrusiones en un sistema SCADA, es posible determinar el nivel de rendimiento y hasta qué punto son efectivas las señalizaciones desarrolladas. Dentro de esos márgenes, la solución detecta satisfactoriamente una serie de ataques típicos y permite la rápida ejecución de cuantas medidas sean necesarias.

6.2 Trabajo futuro

Se desea destacar como posibles líneas de trabajo futuro una serie de mejoras que han quedado pendientes durante la realización de este Trabajo de Fin de Grado:

Optimización del código:

En este área se vislumbran dos posibles actuaciones. La primera es realizar una serie de modificaciones en el sistema desplegado de manera que la ejecución de las diferentes fases se realice en paralelo, optimizando así tiempos y permitiendo que en ningún momento la captura de paquetes se detenga. Esta mejora se llevaría a cabo mediante la generación de diferentes hilos dentro del mismo código, lo cual es posible diseñar sin necesidad de modificar el lenguaje de programación utilizado para el desarrollo de este IDS.

La segunda posibilidad (totalmente compatible con la anterior) es proceder a compilar el código de manera que los tiempos de ejecución se reduzcan y que el rendimiento final en cuanto a tasas de paquetes por segundo que es capaz de gestionar se vea incrementado.

Además se puede aumentar el número de opciones configurables a través de la línea de comandos, como la IP objetivo que se desea monitorizar.

Optimización del algoritmo:

Creación y ajuste fino de variables de media y desviación para las estadísticas generadas, de manera que éstas sean totalmente independientes de las variables actuales y que permitan la personalización de los umbrales de detección en función de las necesidades de cada red y de cada sistema SCADA. Esto mismo es aplicable a las variables de peso en valor definidas.

Inclusión en el código del sistema de ponderación:

La generación una serie de métricas permite tratar de identificar de una manera más precisa el ataque o amenaza que se recibe. Esto se realizaría mediante un sistema de códigos u órdenes que indiquen el nivel de criticidad y relacionar estos valores con el tipo de amenaza recibida.

Establecimiento de un umbral máximo de peticiones por segundo:

En base a los resultados obtenidos en el punto 5.4, una de las líneas futuras más sencillas de implementar sería la de establecer un umbral máximo de peticiones por segundo que indicase un límite inequívoco y permitiese afirmar sin lugar a dudas de la presencia de una amenaza en la red desplegada.

7 Referencias

[1] Ministerio del Interior, «Avance de datos de Cibercriminalidad,» *Secretaría de Estado de Seguridad, Gabinete de Coordinación y Estudios,* p. 5, 2013.

[2] P. O. a. M. Phillips, «Intrusion Detection and Event Monitoring in SCADA networks,» pp. 161,162,172, 2008.

[3] A. U. W. H. Dayu Yang, «Anomali-Based Intrusion Detection for SCADA Systems,» *Nuclear Engineering, University of Tennessee,* 2008.

[4] C.-H. R. L. Y.-C. L. K.-Y. T. Hung-Jen Liao, «Intrusion detection system: A comprehensive review,» 2012.

[5] S. S. Bonnie Zhu, «SCADA-specific Intrusion Detection systems: A survey and Taxonomy,» 2013.

[6] C. L. S. K. Chenfeng Vincent Zhou, «A survey of coordinated attacks and collaborative intrusion detection,» 209.

[7] A. C. M. M. A. T. Igor Nai Fovino, «An experimental investigation of malware attacks on SCADA systems,» 2008.

[8] S. A. L. R. D. W. Vinay M. Igure, «Security issues in SCADA networks,» 2006.

[9] R. C. M. P. S. S. Peter Huitsing, «Attack taonomies for the Modbus Protocols,» *2008.*

[10] Modbus-IDA, «MODBUS Messaging on TCP/IP, Implementation Guide».

[11] B. D. Steven Cheung, «Using Model-based Intrusion Detection for SCADA Networks,» 2006.

A. Anexos

I. Entorno de pruebas

I.1 Sistema virtualizado

En primer lugar, es necesario conseguir un entorno estable y libre de posibles errores en el funcionamiento que no desvíe la atención, empeore o disimule los errores para avanzar correctamente. En un primer momento, la opción más sencilla de configurar con el fin de poder probar ciertos elementos como el servidor y el cliente es instalar cada uno de ellos en un equipo físico diferente. Las herramientas disponibles exigen desde el primer momento que el sistema operativo sea una distribución de Linux, en la que Python funcione adecuadamente y además se encuentren todas las librerías necesarias.

Con estas condiciones en mente se decide desplegar la distribución Manjaro OS en dos equipos, basado en ArchLinux que permite una instalación sencilla y una capacidad de despliegue de librerías sumamente intuitiva. Tras las primeras pruebas se detecta que un entorno de despliegue real dificulta el progreso por las diferentes calidades de tarjetas de red, variaciones en la calidad de la conexión o incompatibilidades con otros sistemas ya instalados en esos equipos.

Con el fin de mejorar el rendimiento se decide trasladar y desplegar dos máquinas virtuales mediante el sistema Virtualbox. Esta configuración se mantiene durante gran parte del desarrollo, realizándose pruebas de conexión que permitan comprobar todas y cada una de las opciones. La arquitectura en este momento es la de un sistema operativo Manjaro OS en el que se despliegan dos máquinas virtuales que actúan como servidor y cliente. A pesar del buen funcionamiento, se decide realizar una mejora que permita asignar mayores recursos a cada máquina virtual.

Tras realizar un pequeño estudio sobre las distribuciones más adecuadas para las pruebas de este estilo, dos de ellas se destacan como las opciones más aceptables para la tarea. A partir de algunas pruebas de transmisión y ante la gran disponibilidad de recursos orientados al análisis de seguridad, se decide la siguiente arquitectura orientada a pruebas:

Sistema Operativo anfitrión: Windows 10, en el que se instala VMWare Workstation 11, y en el que se generan máquinas virtuales con sistema operativo idéntico.

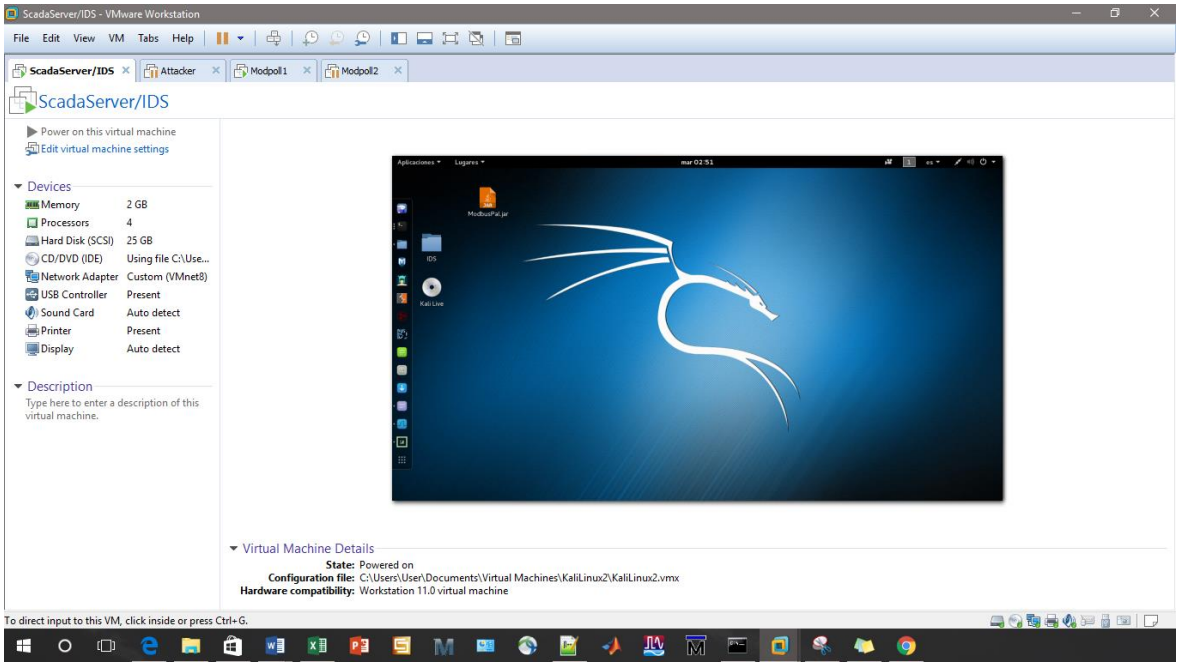


Figura I-1: Configuración de máquinas virtuales en VMware Workstation 11

Máquinas virtuales sobre las que se carga la distribución KaliLinux, conocida por su orientación plena hacia el análisis de seguridad de sistemas informáticos y de comunicaciones. Cada una de estas máquinas recibe una asignación de memoria RAM de 2 Gb, así como acceso a procesamiento en 2 procesadores por 2 núcleos por procesador y 25 Gb de memoria de almacenamiento.

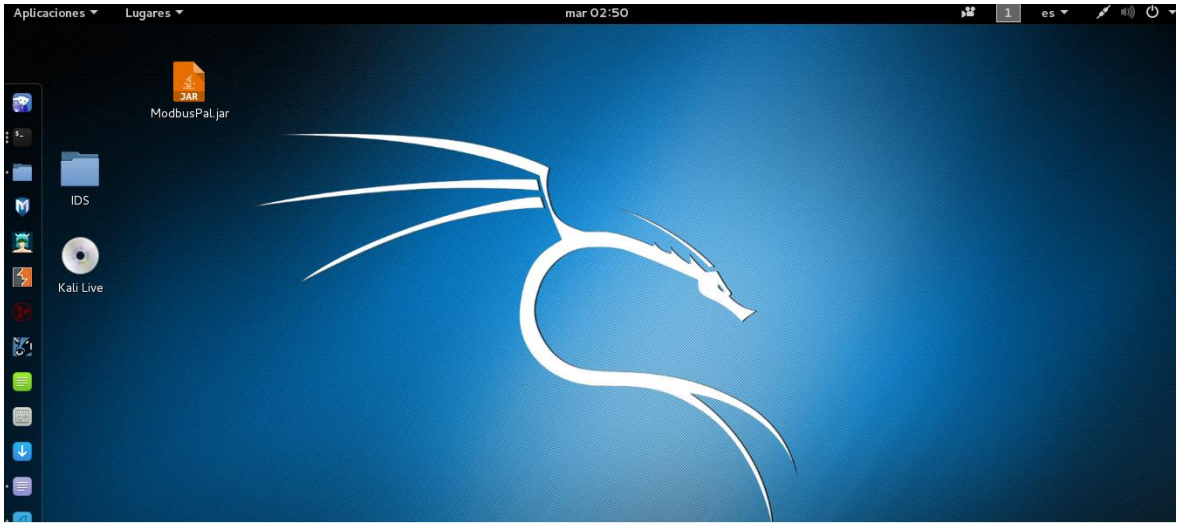


Figura I-2: Escritorio de la distribución KaliLinux 2.0

Cada una de estas máquinas recibe un rol: **servidor, atacante, cliente 1 y cliente 2**, pero en esencia son idénticas a excepción de las ejecuciones que se realizan en cada una de ellas.
A continuación se interconectan las máquinas virtuales y el anfitrión. Esto se realiza mediante un adaptador de red virtual que en este caso se denomina Vmnet8 y cuya configuración es la que aparece en la siguiente figura:

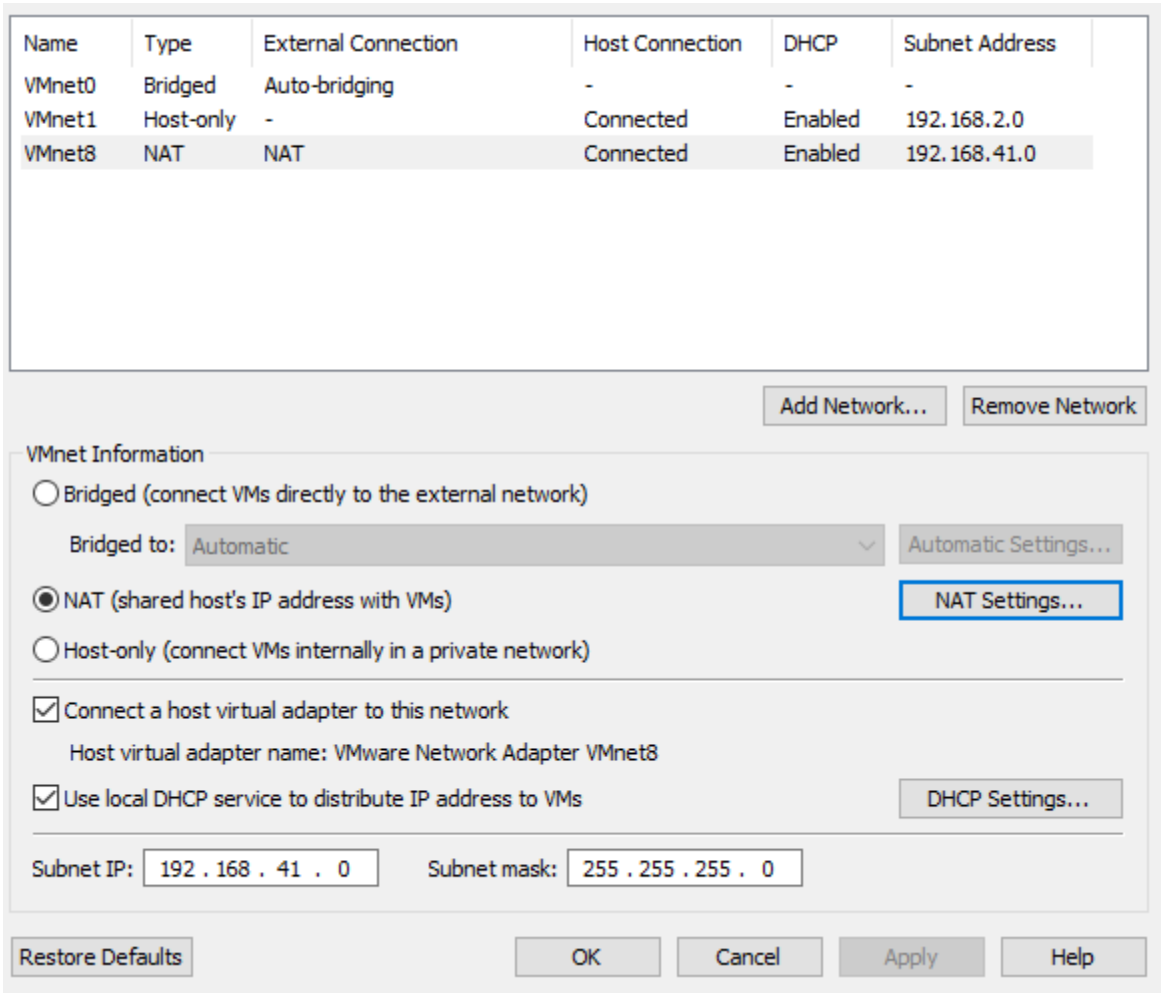


Figura I-3: Configuración de red necesaria para la comunicación entre máquinas virtuales

Todas las máquinas virtuales adquieren IP mediante DHCP de la subred 192.168.41.0 con máscara de subred 255.255.255.0 y tienen salida a internet mediante NAT con la IP del equipo anfitrión. De igual manera, el equipo anfitrión toma el rol de Gateway de esta subred con la IP 192.168.41.1.

Una vez realizada esta tarea previa de interconexión, queda configurar los extremos de la conexión y realizar las pruebas.

I.2 Configuración de ModbusPal Server

Una vez iniciada sesión en KaliLinux con rol de servidor, únicamente es necesario ejecutar en el terminal el siguiente comando

```
root@sec:~/Escritorio# sudo java -jar ModbusPal.jar
```

Figura I-4: Comando de ejecución de ModbusPal

Una vez ejecutado correctamente, nos aparece la siguiente ventana en la que únicamente es necesario pulsar “Learn” y a continuación “Run” antes de poder empezar a trabajar con el servidor ModbusTCP.

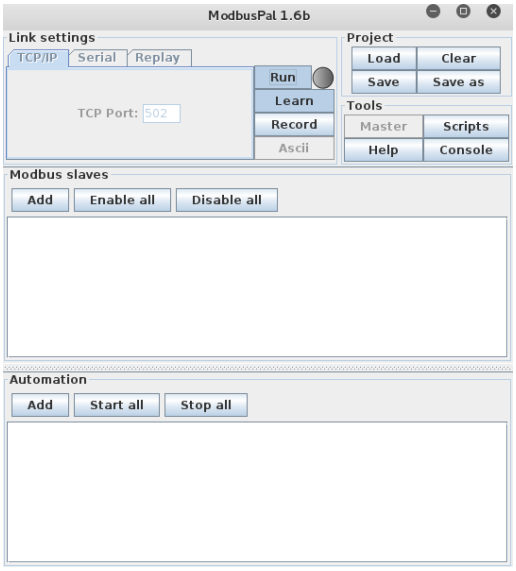


Figura I-5: Interfaz gráfica de ModbusPal

El siguiente paso es ejecutar el cliente en otra máquina virtual.

I.3 Ejecución del Master Modpoll (Cliente ModbusTCP)

El primer paso para ejecutar este programa es darle permisos de lectura, escritura y ejecución en Linux mediante el comando "chmod". Modpoll permite obtener datos del servidor de muchas maneras, pero en este caso concreto interesa ejecutar lo siguiente:

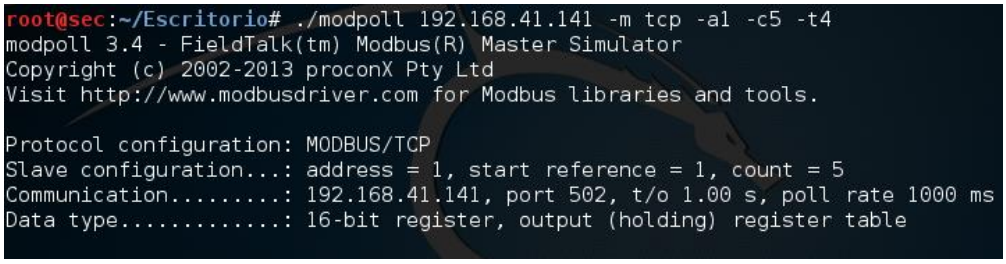


Figura I-6: Ejecución del cliente Modpoll

Esta ejecución especifica en orden de introducción de opciones: dirección del servidor, conexión ModbusTCP, ID de la unidad de la que se desean obtener datos, número de registros a obtener y finalmente, formato numérico de los datos. Dado que el ModbusPal se encuentra en modo "learn" esta configuración es trivial.

I.4 Generación de tráfico no legítimo

Se usan módulos ya disponibles de Metasploit, Nmap y otras herramientas, así como la herramienta de generación de paquetes "PackETH". De este conjunto de herramientas se obtienen las pruebas que más tarde se muestran. Como ya se ha comentado, en los dos primeros casos se utilizan módulos o comandos ya desarrollados, mientras que en PackETH se opta por crear un paquete con un formato similar al transmitido habitualmente en la red y replicarlo en función de las necesidades de cada prueba.

II. Compendio de pruebas realizadas

II.1 Pruebas extra con NMAP

nmap -v -sT 192.168.41.141 --data-length 1000 -p 502

A continuación se genera un nuevo escaneo de tipo "Connect()", que como se observa genera un mayor número de alertas, siendo especialmente destacable respecto al anterior escaneo la presencia de TMB_ALARM, que indica que el ratio de paquetes TCP/ModbusTCP ha sido alterado.

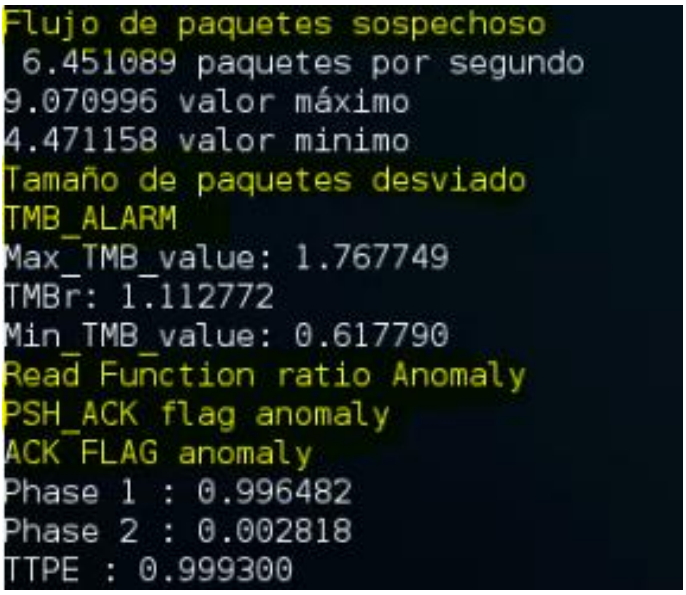


Figura II-7 : Respuesta del IDS a nmap -v -sT

En la escala generada este ataque recibe por tanto una mayor criticidad, debido a la presencia de esta señal que se ha comentado anteriormente. Su nivel de criticidad aumenta, situándose éste en 10.5.

Este caso de amenaza generada mediante Nmap es de los que mayor grado de criticidad provoca en el IDS desarrollado.

En el anexo de pruebas se encuentran algunas de las capturas de ataques realizadas de manera adicional, con el fin de detectar el nivel de incidencia que el tamaño de datos transmitidos por paquete transmitido. La conclusión es que el IDS desarrollado sigue siendo efectivo ante intentos de conexión con paquetes más pequeños.

nmap -v -sM 192.168.41.141 --data-length 1000 -p 502

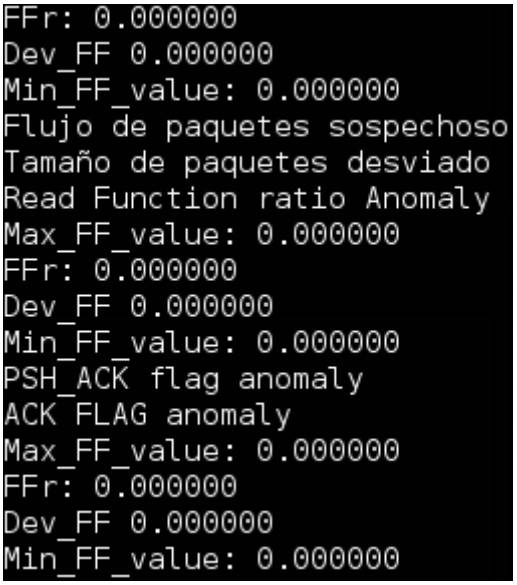


Figura II-8 : Respuesta del IDS a nmap -v -sM

De nuevo se prueba con otro escaneo diferente, que en este caso genera un índice de criticidad de 9. En la siguiente figura se observa la detección del flujo anómalo mediante la presencia de 5 de las señales de alerta. Como habitualmente, las dos primeras corresponden con una desviación o anomalía en el flujo de paquetes por segundo y en el tamaño medio de los paquetes dentro de ese mismo segundo.
Igualmente, al tratarse de un escaneo de puertos la función que lee del paquete no se corresponde con las habituales y provoca una desviación del habitual valor. Las banderas TCP igualmente indican un cambio en el tipo de paquetes recibido que no encaja con la transmisión canónica de ModbusTCP.

II.2 Pruebas extra con Metasploit

En este apartado del anexo se adjuntan la cuarta prueba realizada con Metasploit, con resultados similares a los casos anteriores.

Se tratan de un “discovery” que trata de conseguir la información de relevancia del servidor de la red. Las opciones a configurar son las que aparecen en la siguiente figura:

```
msf auxiliary(modbus_findunitid) > use auxiliary/scanner/scada/modbusdetect
msf auxiliary(modbusdetect) > show options

Module options (auxiliary/scanner/scada/modbusdetect):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    502              yes       The target port
   THREADS  1                yes       The number of concurrent threads
   TIMEOUT  10               yes       Timeout for the network probe
   UNIT_ID  1                yes       ModBus Unit Identifier, 1..255, most often 1

msf auxiliary(modbusdetect) > set RHOSTS 192.168.41.141
RHOSTS => 192.168.41.141
msf auxiliary(modbusdetect) > run

[+] 192.168.41.141:502 - MODBUS - received correct MODBUS/TCP header (unit-ID: 1)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figura II-9: Opciones de configuración para modbusdetect

Las alertas disparadas por la ejecución de este módulo son las siguientes:

```
Dev_FF 0.000000
Min_FF_value: 0.000000
Max_FF_value: 0.000000
FFr: 0.000000
Dev_FF 0.000000
Min_FF_value: 0.000000
Flujo de paquetes sospechoso
Tamaño de paquetes desviado
TMB_ALARM
Read Function ratio Anomaly
Max_FF_value: 0.000000
FFr: 0.000000
Dev_FF 0.000000
Min_FF_value: 0.000000
PSH_ACK flag anomaly
ACK_FLAG anomaly
Max_FF_value: 0.000000
```

Figura II-10: Respuesta a módulo modbusdetect

II.3 Resultado mediciones de tiempo de ejecución

Con el fin de comprobar el funcionamiento en tiempo real y las posibles pérdidas de paquetes producidas por la incapacidad del sistema de gestionar cierto número de peticiones se incluyen en el código una serie de marcas o señales de tiempo que ayudan a identificar y a tratar de optimizar el tiempo emlpleado en cada una de las tareas.

Dada la naturaleza secuencial de las fases desarrolladas. Estas deben sumar una cantidad similar al segundo por cada una de las ejecuciones de código completo que se realice. En una evolución futura en la cual se empleasen diferentes procesos o hilos para el funcionamiento simultáneo, dichos intervalos no tendrían la necesidad de sumar una cantidad similar al segundo en alguno de los casos, ya que la fase de captura de paquetes podría indicar el tiempo necesario para el almacenamiento temporal del paquete.

A continuación se muestran los tiempos para diferentes ejecuciones de código. En primer lugar se muestra la ejecución para una transmisión habitual, en segundo, los tiempos empleados en una transmisión que implique una conexión legítima simultaneada con un intento de ataque y finalmente se realiza una prueba de estrés de estos tiempos.

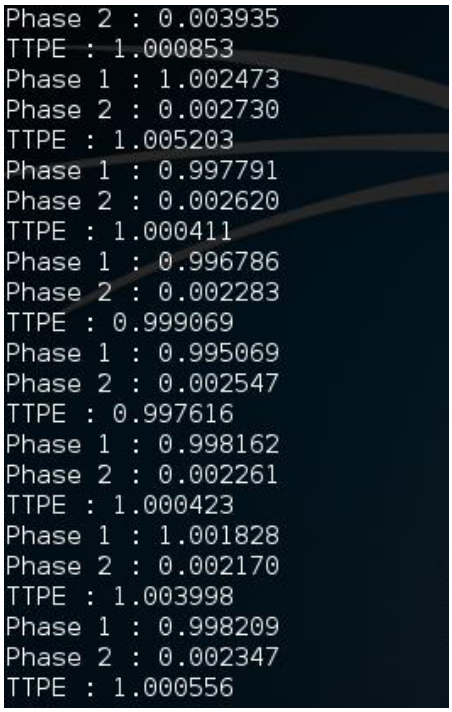


Figura II-11: Tiempos de ejecución por fases y total

En esta primera captura se observa que el tiempo real de ejecución provoca un retardo máximo en la ejecución del orden de los 0,5 milisegundos. Teniendo en cuenta la tasa de 3 paquetes por segundo, dicho retraso es despreciable.

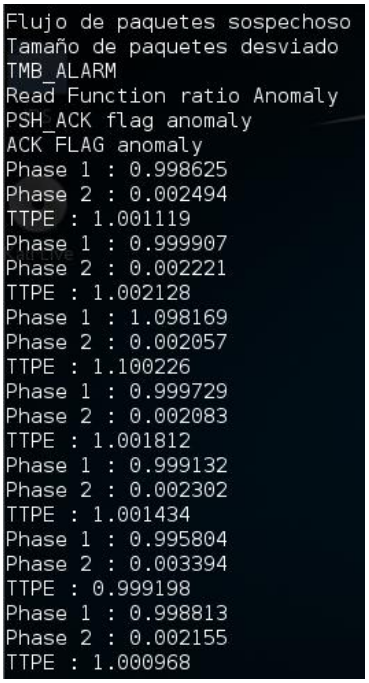


Figura II-12: Tiempos de ejecución con presencia de ataque sencillo

Ante un ataque sencillo, el sistema sufre retrasos de una décima de segundo de los que se recupera en la siguiente iteración. La normalización absoluta parece llegar a los dos ciclos de producirse el retraso. Dado un retraso de una décima, las probabilidades de afectar a los cálculos de variaciones anómalas de flujo quedan muy reducidas y son prácticamente despreciables por la baja tasa de transmisión del servidor SCADA. Además, queda reducida su importancia una vez observado el "output" del IDS ante otra de los módulos de metasploit, en este caso "modbus_findunitid", el cual realiza peticiones de una manera más constante.

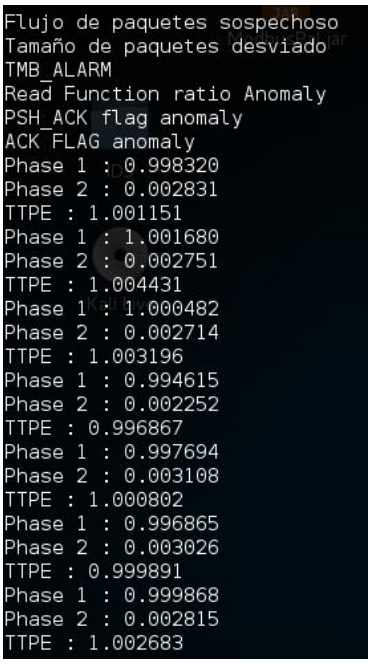


Figura II-13: Tiempos de respuesta ante modbus_findunitid

Finalmente se realiza la prueba de estrés a una tasa cercana a la máxima aceptable antes de comenzar la pérdida de paquetes.

Se configura packETH para el envío a una tasa de 1 Mbit/s con el fin de provocar una velocidad de ejecución alta en el IDS, tal y como muestra la siguiente figura.

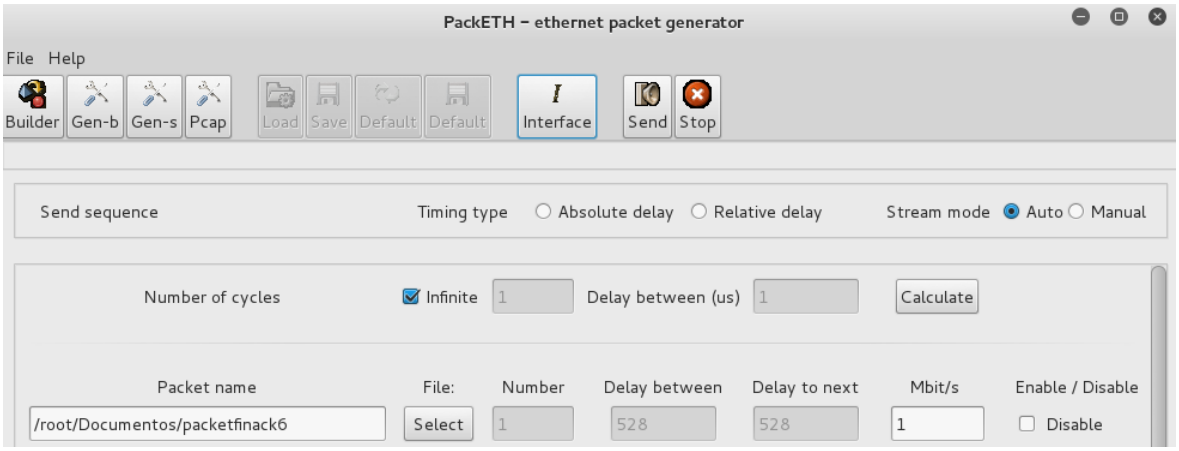


Figura II-14: Ejecución de packETH a 1 Mbit/s

Y la respuesta a tal estímulo es la siguiente:

```
Flujo de paquetes sospechoso
Tamaño de paquetes desviado
TMB_ALARM
PSH_ACK flag anomaly
ACK FLAG anomaly
Phase 1 : 0.928085
Phase 2 : 0.305727
TTPE : 1.233812
Flujo de paquetes sospechoso
TMB_ALARM
PSH_ACK flag anomaly
ACK FLAG anomaly
Unusual Flag anomaly
Phase 1 : 0.626801
Phase 2 : 0.289667
TTPE : 0.916468
Phase 1 : 0.700913
Phase 2 : 0.320289
TTPE : 1.021202
Read Function ratio Anomaly
Phase 1 : 0.662883
Phase 2 : 0.276789
TTPE : 0.939672
Phase 1 : 0.668152
Phase 2 : 0.334589
TTPE : 1.002741
```

Figura II-15: Respuesta a estímulo a 1 Mbit/s

Durante la ejecución de esta prueba se observan cambios en el ritmo de ejecución del sistema, lo cual podría indicar una posible pérdida o recepción ilícita de paquetes. Dada la estabilidad habitual, esto también podría contribuir a identificar un posible ataque.